COMS E6156
Topics in Software Engineering

May 10th, 2016

Final Report

team Review Changes World
Ruoqi, Wang (rw2612@columbia.edu)
Yuyang, Liu (yl3319@columbia.edu)
Chen Yu (cy2415@columbia.edu)

Introduction

For ages, researchers in the field of requirements engineering have been envisioning a model that could enable mass participation of users when designing a software, and a set of tools where developers could use to analyze the data user created. Based on existing discussions[1] and techniques accessible to us, in this project we would like to actually implement a model with powerful enough analytical capabilities where developers and users could interact and work together for better releases (for example, user may post requests, comments and suggestions on the application they use, rate particular comments published by other users, or even join the debate over one specific feature). Different from existing services, our projects would incorporate more mechanisms for user participation (other than merely casting votes), stay fully customizable to every developer using the platform and provide built-in analytics to people interested. Hopefully, this would provide better requirements analysis to developers eager to improve their applications.

Deliverables and Access Methods

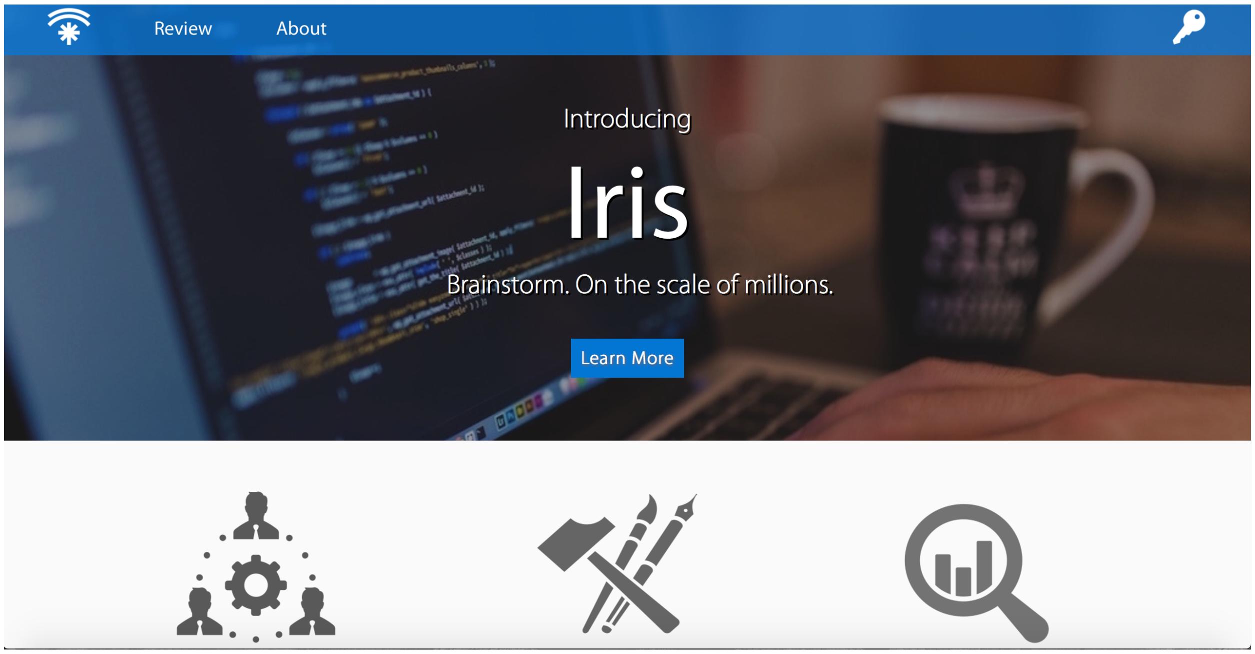
Deliverables	Access Methods
Source Code	Hosted on Github Accessible at https://github.com/michaelawyu/Iris-Class-Project-
Web Application	Hosted on Microsoft Azure Servers, Accessible at reviewchangesworld.eastus.cloudapp.azure.com:5000 *Server under Maintenance from Midnight May 10th, 2016 to 8:00 AM May 11th, 2016*
iOS Client	Compile-able through Xcode
restfulAPI	Delivered as a part of Web Application, currently hosted on Microsoft Azure Servers *Server under Maintenance from Midnight May 10th, 2016 to 8:00 AM May 11th, 2016*
Final Report (Tutorials Included)	Available on Courseworks

[1] For example, see Johann et al.'s *Democratic Mass Participation of Users in Requirements Engineering?* IEEE Software.

Tutorials

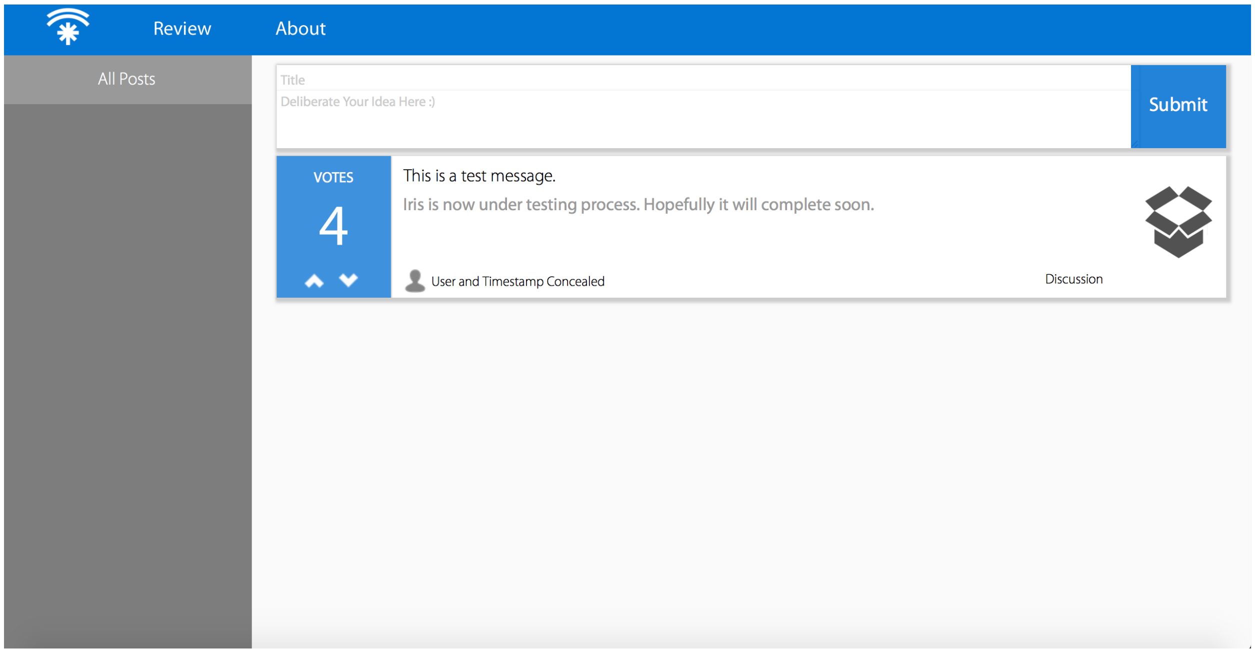
In the Web Application

Web application is hosted on Microsoft Azure servers at reviewchangesworld.eastus.cloudapp.azure.com:5000. The index page looks as the follows:



Use the *key* icon to login or register an account. You may also want to use *consumer_testaccount (*Password: *consumer)* or *dev_testaccount (*Password: *dev)* to try things out.

Users may use the *Review* section to see all the feature requests currently in the database. The controls available in the *Review* section (for example, the up vote button) are determined by the configurations of developers in charge of current product. Each feature request also comes with a discussion portal for interaction among users.

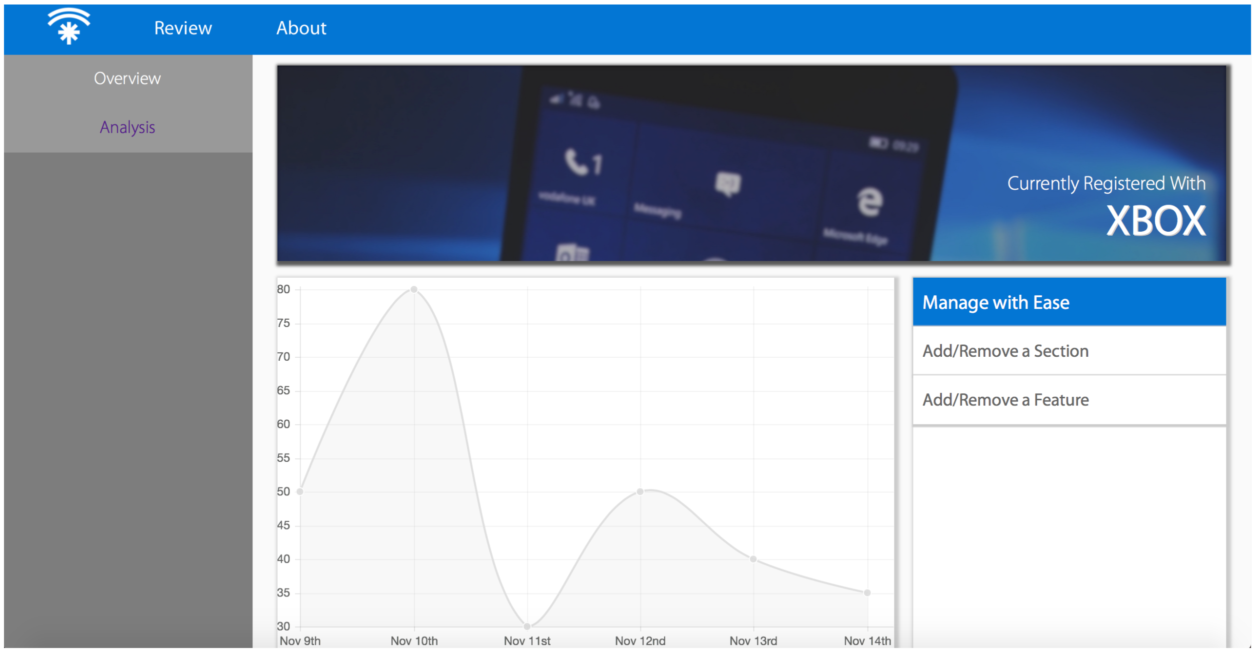


This is a test message.

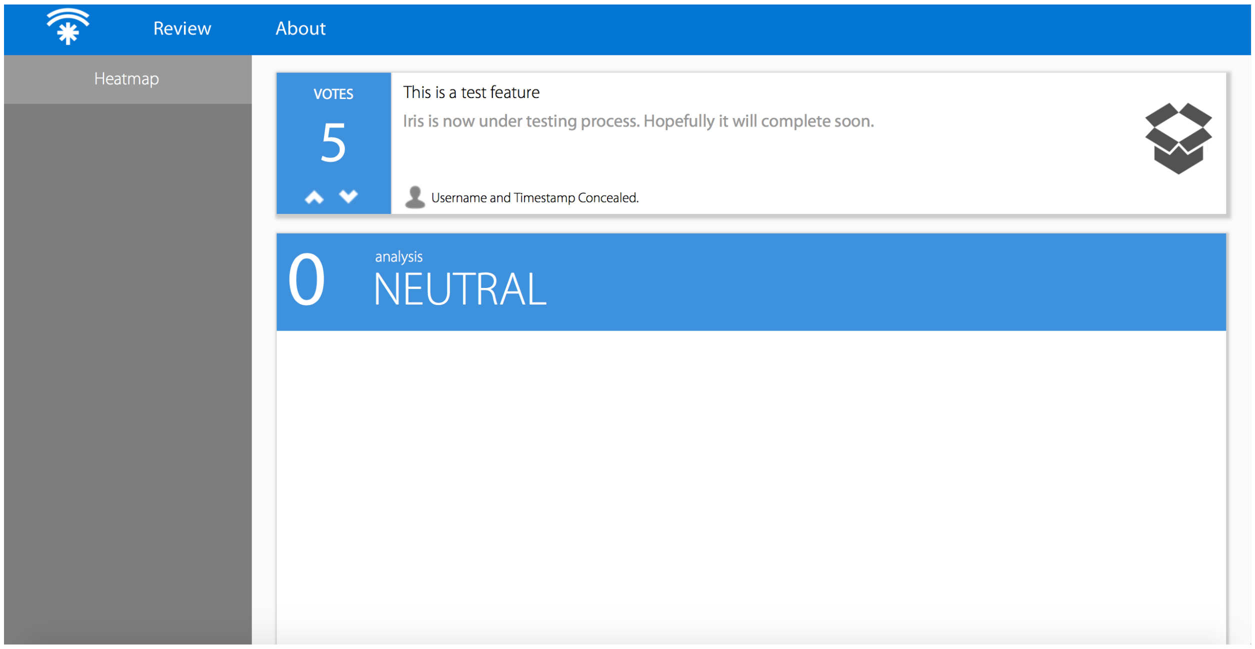
Iris is now under testing process. Hopefully it will complete soon.

User and Timestamp Concealed

If currently logged in as a developer, a *Control Panel* section would be available at the top. Developers may use this page to track the progress of their product and change configurations in the system.



Analysis tab on the left panel is for NLP (Natural Language Processing) analyses. We currently offer two separate tools: *Topic Extraction* and *Sentiment Analysis*. The former is for extracting keywords for all the feature requests and the latter helps discover features request users love most (or hate most).



This is a test feature

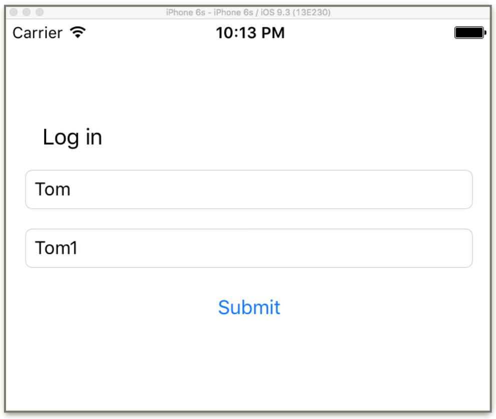
Iris is now under testing process. Hopefully it will complete soon.

Username and Timestamp Concealed.

0 analysis NEUTRAL

iOS Client

iOS client offers a “compact” version of our service: user may log in, view sections/feature requests and vote for (or against a feature).

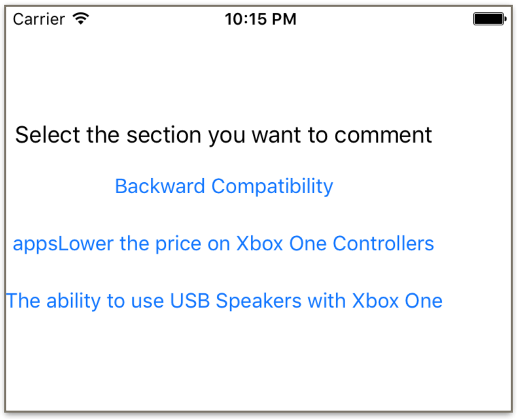


Log in

Tom

Tom1

Submit

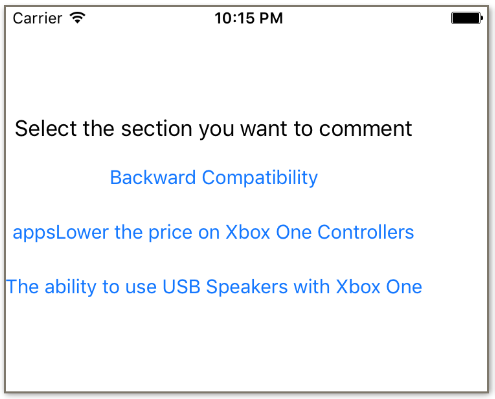


Select the section you want to comment

Backward Compatibility

appsLower the price on Xbox One Controllers

The ability to use USB Speakers with Xbox One



Select the section you want to comment

Backward Compatibility

appsLower the price on Xbox One Controllers

The ability to use USB Speakers with Xbox One

External Softwares Referenced

Component	Developer and Access Methods	Functionalities
Flask framework (Werkzeug and Jinja2 come as a part of Flask)	Armin Ronacher and others Project available at http://flask.pocoo.org Source Code available at https://github.com/pallets/flask *Open Source, Distributed under BSD License*	Foundation of Web Application
Xcode + Swift	Apple Available at https://developer.apple.com/xcode/ *Commercial, Proprietary Software*	Development of iOS Client
ntlk	NLTK Project Available at http://www.nltk.org *Open Source, Distributed under Apache License 2.0*	A Part of NLP Analysis Component
Stanford Topic Modeling Toolbox	Stanford University Available at http://nlp.stanford.edu/software/tmt/tmt-0.4/	A Part of NLP Analysis Component
Chart.js	Chart.js Project Project available at chartjs.org Source Code available at https://github.com/chartjs *Open Source, Distributed under MIT License*	Charts and Diagrams in Web Applications
PostgreSQL	The PostgreSQL Global Development Group Available at http://www.postgresql.org *Open Source, Distributed under PostgreSQL License*	SQL Database in the Backend

Component	Developer and Access Methods	Functionalities
SQLAlchemy	SQLAlchemy Development Group Available at http://www.sqlalchemy.org *Open Source, Distributed under MIT License*	Communication between Server and Database
	Web Application and restfulAPI developed in Python *iOS client developed in Swift*	

Results

Our submission implemented the following functionalities:

- Web Application
  - User may register as *consumer* or *developer*
  - *Consumer* may post a feature request, vote for (against) a feature request and interact with developers and other users
  - *Developer* has the option to configure how consumer would be able to interact with the system, including the ability to up vote (or down vote), the total number of votes available, availability of posting feature requests and follow-ups.
  - *Developer* may set up different sections for different categories of feature requests
  - Two NLP analytical tools, *Topic Extraction* and *Sentiment Analysis* are offered for better insights into the data

- iOS client
  - User may log in to the system
  - User may view the sections available and choose from a collection of feature requests
  - User may vote for (or against) a feature, just like what they could do in the web application
  - Client interact with the server through HTTP requests
  - Repetitive votes are not allowed

- restfulAPI
  - Listening for HTTP requests sent by the client, accessing the database and fetching the result

- Other Infrastructure
  - a PostgreSQL database is set up for data storage and access.

- Differences with *UserVoice.com*
  - More customization tools offered to developers
  - Built-in NLP analytical tools

Things Learned

from Ruoqi, Wang

During this project, I have learned how to develop an ios application using swift which is an area that I have never touched before. Swift is a safe, fast and interactive programming language for IOS, OS X, tvOS, watchOS. The structure of this application uses MVC (Model-View-Controller) model. The UI interface of the application is described in storyboard. The ViewController class controls the all the action of a specific view. For each ViewController object in the storyboard, a ViewController class is need in swift to define all the actions to perform when something changes in the storyboard. Since we need more control over the user's' request, such as perform some custom authentication, the NSURLSession is used to establish the connection to the HTTP Servlet and have some interaction between the server and application. After the user have entered something in the app, it will send the data to the database in the server to find a match or do a simple query, and send back the corresponding results to the app. The data sent back is parsed into meaningful sections in order to display properly on the screen. Besides, one thing that draws my attention is all the wrapped conversions in swift. The term "wrapped" means we should be careful in the situation that optional variables as a present, wrapped in shiny paper, which might be empty. And it is a usual problem when I do programming in swift.

from Siyu, Liu

I was responsible for the front end and client side in some of my past projects. So in this project, I decide to work on the server side to gain more experience. My part includes database creation, data interaction and management between the server side and client side(web terminal and IOS terminal), user authentication procedure, and provides RESTful API for the client side to use. In this project, most valuable things I have learned is the how to use the Flask and its RESTful functionalities. Flask is a framework aimed at small applications with simpler requirements. In my perspective, the most typical feature of Flask is simplicity and it's easier for us to understand the intent of the code. With the help of Flask extension, I have built RESTful API quickly, and it is convenient for the front end and back end to make interaction. Futhermore, flask-SQLAlchemy makes it more convenient when compared with plain SQLAlchemy, we do not have to specify tablename or define a scoped session. And I could manage the data transferred between the server and client more efficiently with all of these features. Convenience is the biggest advantage of flask, and I think this experience will help me in using API in Flask and do programming more efficiently in my future project.

from Chen Yu

I built the web application and the NLP analysis component in this project. It is inspiring and help to use what I have learned this semester in the NLP course to a actual project, and compiling different parts together as a whole is always a fulfilling experience. I had experience with Flask framework before, though this project still helps me find out a few advanced features that I do not know before. The most important thing I have learned in this project however is not technical; for our project working together as a team efficiently is always a challenge, and I feel that I am a little better informed what development in real world would feel like.

Incomplete Parts

Some features, envisioned when we started the project, are not built into the final submission, including delegation, weighted vote and a fully developed request feature status system. iOS client was planned as a full-fledged utility with the same functionalities as the web application, yet we felt it difficult to accomplish in current time window. It is evident that NLP analysis component could do better classifying interactions between developer and users, but training such a model apparently goes out of our capabilities for now.

Challenges

From a technical perspective, as we used a collection of external components in our project, none of us is master of them all. It took us a while to figure some components out as a team, and making them work together as a whole takes even more time. It is a valuable experience for us all.

The bigger challenge lies in the less technical part. The features we have in mind have dependencies on each other (for example, iOS applications relies heavily on the availability of restfulAPI) and it sometimes is indeed difficult to get everyone on the same page, as we each understand the project in a slightly different way in the beginning. We did try out a small number of techniques we have learned in the software engineering class (the estimating poker game, burn-down chart, just to name a few), unfortunately like many things in the world, the way you operate things in book is not necessarily the same as the way it works in real-world. It is helpful to see those techniques in action and we do wonder how we could find a better way to work as a team, with the help of knowledge in software engineering.

Last but not least, our project is completed in a short time window. Given more time, we could come out with more features and conduct a few more system tests to eliminate potential bugs. There still remains many to discuss in the field of requirement engineering.

Responsibilities of Team Members

Member	Responsibilities
Ruoqi, Wang (rw2612@columbia.edu)	Development of iOS Client
Yuyang, Liu (yl3319@columbia.edu)	Development of restfulAPI Database Administration
Chen, Yu (cy2415@columbia.edu)	Development of Web Application NLP Analysis Component
	Final Report and other Miscellaneous Files completed collectively.

Discussion

Regarding Adaptive Design Problem

As discussed earlier today during the demo, our project currently “freezes” data in the server when developer proposes a configuration change in their own project. Adding more adaptability to the design as to accommodate frequent changes often occur in the course of development would definitely make our project more useful, and we are looking forward to exploring that option if possible in the future.

Regrading Hierarchy of Roles in the Project

Currently there are two roles available in the project: developers and users. We unanimously agree that adding a hierarchy of roles to the project (say, system-wide administrator, organization-wide administrator) would cater better to the real-world needs, and we do hope to have the opportunity to build it.