QM Lab Course Project

Design Document

Group 3

Angela Stankowski (ans723), Bengin Lee (bel529), Connor Nettleton-Gooding (cwn973), Corey Hickson (crh208), Darvin Zhang (ddz369), James McKay (jlm012), Jordan Wong (jtw289), Mack Mahmoud (mtm464), Matt Hamilton (mch986), Michael Kelly (mlk121), Michael Ruffell (mar492), Mitchell Lau (chl929), Royce Meyer (brm979), Shane Williamsom (saw056)

February 7th, 2016

I. Introduction

II. Architectures

For our system, we are hoping to employ a number of architectures. Overall, we would like to have a broad architecture which we can use to describe generally how our system works. This section will look at different architectures and how we might employ them.

3-Tier Architecture

The most common architecture for a web application would be the 3-tier architecture, with a front end client that handles all of the visual and interactive aspects that users would handle, with a application tier handling all of the collaboration and ensuring each client is updating to the most recent version of the document, and a backend storing and managing all of the data we choose to keep (currently undetermined).

2-Tier Architecture

A 2-tier architecture would effectively host our entire application on the client side with minimal interaction happening with the database, except for storage. Our complete from end client would send relevant data back to our backend as needed.

Server-client

Alternatively, we could setup a client side application which would run the bulk of the work, while occasionally interacting with the server side. This would be very similar in style to the 2-tier architecture.

Model-view-controller

Lastly, is a variation on the model-view-controller. The primary focus of the application would be the view and its functionality, while being controlled in the back with a controller, and storing relevant information in our model as needed. The benefit of this would be possible greater utilization of the model, as we could break apart the actual app and have state information in the model.

A. Decision Matrix

To help determine which architecture we should pursue, we created a decision matrix as seen below.

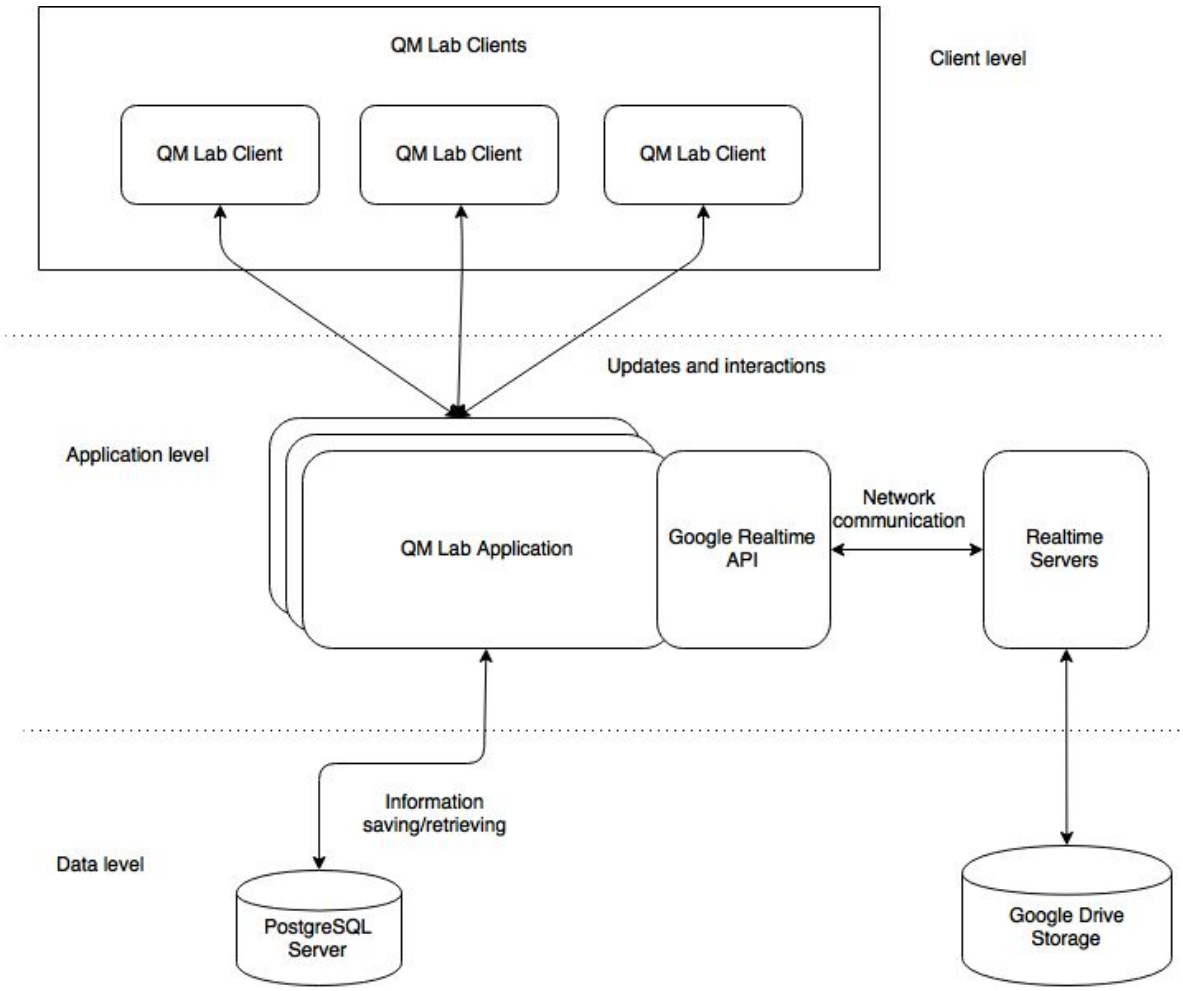
				Rankings					
	Applicability	Modularity	Ease of Implementation	Adaptability	Collaborability	Useability	Security	Testability	Total
Weight (1->5)	4	2	2	4	5	4	3	5	
Architectures (1->10)									
3-Tier Architecture	8	7	7	8	8	9	9	8	235
2-Tier Architecture	7	6	8	6	6	8	6	5	185
Server-client	6	5	6	6	5	5	7	5	161
Model-View-Controller	6	8	7	7	8	8	5	8	209

Decision matrix of different architecture possibilities for our system

We choose to make our matrix based off of applicability, adaptability, usability, and security to meet stakeholder requirements. We choose modularity, ease of implementation, collaborability, and testability to meet the team's requirements to have something we would be successful in using.

From this diagram, we can see that a 3-tier architecture comes out as the most likely to succeed architecture for our overall system. Considering the diagram, as well as our based on our research, we have decided to go forward with a 3-tier architecture for our overall system.

B. Architecture Diagrams

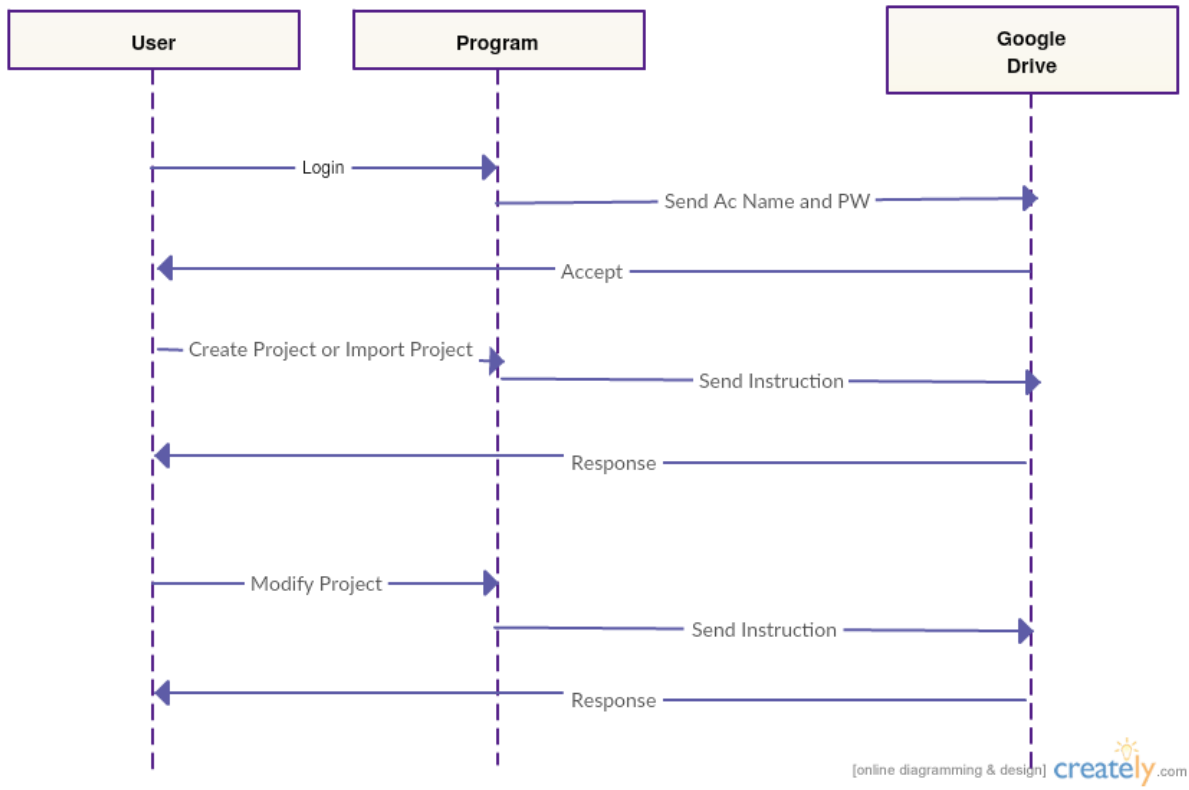


III. Models

interface data structure
pizza
flyweight design pattern (easier to manage, similar to excel)
singleton
publish subscribe observer

IV. Classes

V. Sequence Diagrams



VI. Time Estimates

require documentation	7 hrs
design document	13 hrs
user manual	8 hrs
Test Planning	10 hrs
Testing	45 hrs
Debug	30 hrs

Implementation must have:	10+10+8+5+8+8 = 49 hrs
Support multiple users (OT)	10 hrs

To create, save, and import	10 hrs
Support system dynamic diagram ,stock diagrams,agent ,picture ..... and text	8 hrs
one of a small number of shapes and images	5hrs
Create variables and links between components	8 hrs
A link for the project that for sharing	8 hrs

Implementation should have:	10+10+5+5+5= 35 hrs
A login system	10 hrs
A log of who edited what at certain points - potentially save state/backups	10 hrs
Power to select a set of the diagram	5 hrs
Ability to resize and color the elements for a diagram	5 hrs
Change the font/style of text within components	5 hrs

Implementation could have:	7+10+20+20+5+13+5= 80 hrs
Customized components such as a user specified picture is added and named to the elements available for a certain project/team	7 hrs
Customizable overlay for the software	10 hrs
A chat system - private chat	20 hrs
General templates to guide users	20 hrs

Differents styles of components (different looks available for say a UML class diagram)	5 hrs
Undo function possibly based on history	13 hrs
Support of video maybe a slideshow/add pages to a project?	5 hrs

VII. Conclusion