Team 4 – Bug Overflow

Midterm Paper

Brian Curtis -

Project Manager:

2017

**Summary**

Our project is to create a question and answer website similar to StackOverflow. Our site, BugOverflow, is focused on insects, rather than coding errors.

**Development Methodology – Second Way**

The most accurate description of our development process would be Gene Kim’s “Second Way” of product development. Our group is small, so a right to left feedback loop best describes how we send and receive feedback between ourselves about our progress. This process gives a more meaningful understanding of what is working, and what needs improvement.

**Database platform – MySQL**

MySQL is currently an industry standard. This means that the technology has been proven to work, there is plenty of support for it, and it is compatible with many systems. The main alternative to MySQL would be NoSQL. NoSQL has advantages over MySQL, mainly because its simpler and its table-less, meaning that its lighter. In fact, many large database websites such as Facebook and Google use NoSQL. For the time being as a new site, we have decided to go with what we know, and with what is a standard.

**OS and application stack –Ubuntu 14.04.5, MySQL Workbench, Vagrant, Packer, Apache**

Our team works primarily on Windows operating systems, so it made sense to develop on there. Our team’s database is developed and ran using MySQL workbench. The reason for using MySQL was explained above. Workbench is a free program that our development and infrastructure team was familiar with using. Vagrant and Packer are used to both copy and deploy exact copies of our virtualized servers so that each team member is working with the same setup and is able to deploy the site themselves. Xamp is used strictly for quick testing of server elements, such as our PHP code. Apache is used so that windows can properly view development PHP code. While we work on Windows machines, our actual servers run on a virtualized version of Ubuntu 14. Ubuntu is a lightweight and powerful version of Linux (Unix based) that can be easily virtualized and copied. Amazon Web Services (AWS) is planned to be used to store photo data, as well as host the servers.

**Programming languages/ frameworks – HTML5, CSS, PHP**

Our main code is PHP. PHP is a powerful language capable of interacting with SQL code, as well as managing HTML and CSS. Authentication, data, and site structure will all be managed with PHP. Along with PHP, most of the actual frontend page is still HTML5, which is the internet standard for displaying websites. HTML5 allows us to structure the site to be seen. The design of the website is handled with CSS, which is the only standard for visually structuring websites. Without CSS, the site would be a visual mess of text that is unappealing and confusing to the user.

V**alue Stream – below**



This diagram shows the basics of how we have planned the site to work. From the frontend UI on the left, we move towards the storage and retrieval of data. This data management is crucial to how we display posts from users. The site interacts with the database to grant permissions to varying levels of user privileges, and store user, post, and image information in the right places. Geolocation is a special feature that is mainly important for the user uploading of photos, which interacts with the Google Maps API. All of that comes around to the main storage of data, whether its creating users, creating content, or having the AWS store photos. Overall, this plan is to allow users to go through a line of security (Authentication), then let them upload photos with questions, answer questions, and be able to retrieve that data.

**Cost of Servers – Azure ~$650; AWS ~$400**

There are two main offsite server hosting options available to us: Microsoft Azure’s services, and Amazon Web Services. Our website would require 3 virtualized servers, daily tech support, and 3 additional storage disks. Both provide cloud based servers, and customer support.

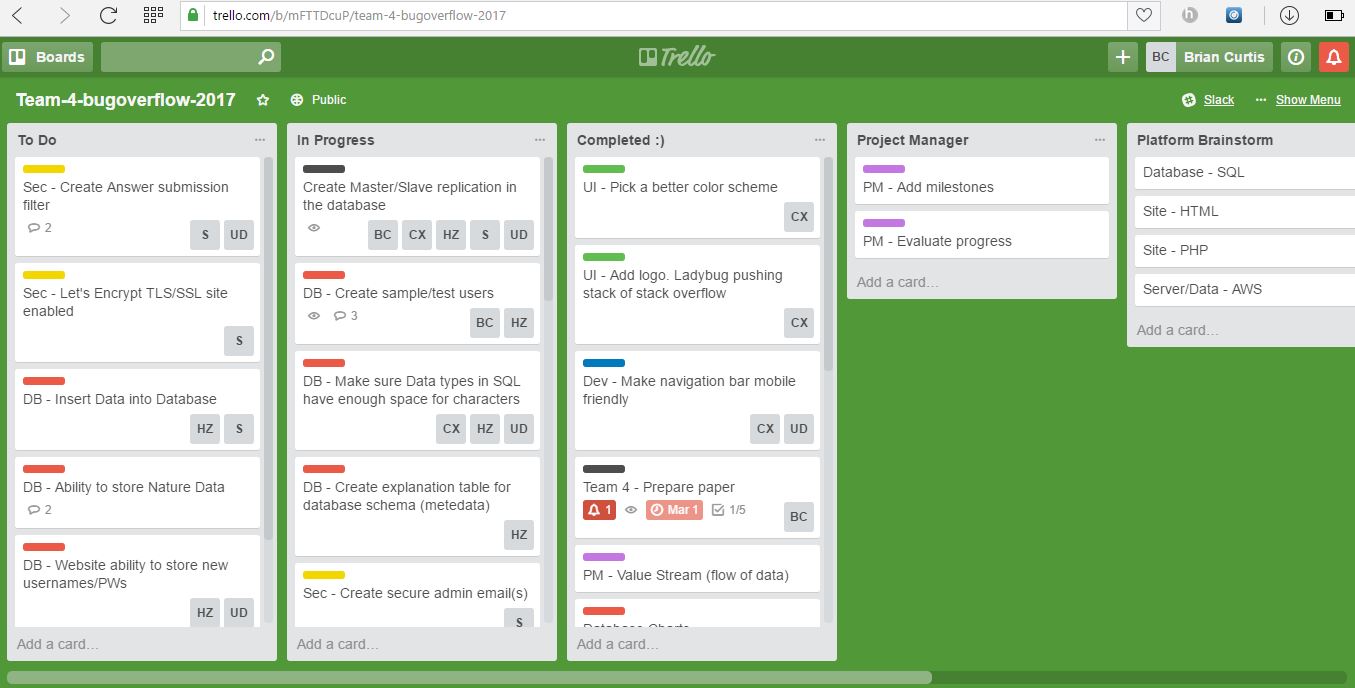
Using Azure’s services would cost an estimated $650 a month to use for 3 A3 servers, with business-level tech support, and 3 managed disks. Each server is about $180, and each managed disk is $20 a month.

AWS would cost about $450 for 3 servers, 2 of which would be specialized as databases, along with business tech support within 1 hour. These 3 servers would be AWS t2.xlarges, meaning that they have dual cores with 8GB of RAM for $130 a month. These machines are estimated as using 100% up-time. There is a storage device of 150GB for storing our SQL data and website. This estimated plan includes in and out data transfer of 120GB a month, as well as developer level tech support. This support includes a 1 on 1 AWS specialist to help with problems in the cloud service for $29 a month.

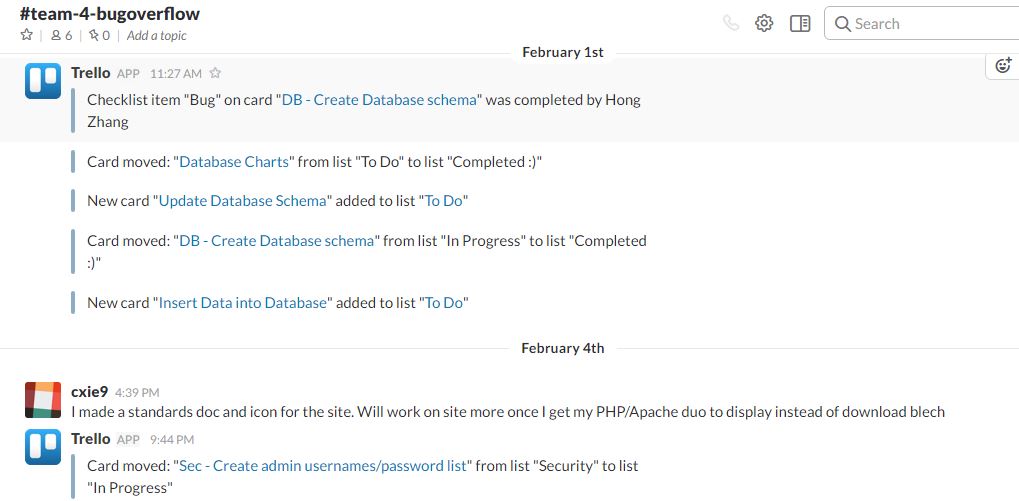
These estimates are very conservative, and based loosely on the statistics of Stackoverflow. It is impossible to know how popular or how much use or website would get, which makes estimating traffic use difficult. Research did show that, if StackOverflow used AWS or Azure to run their site they would have to pay a lot of money, so they manage in-house servers. More research should be done to discover exactly which service would best suit our needs, but AWS has a better initial offer, as well as being a much larger cloud-based operation.

**Project Communication and Tasking**

The Bugoverflow team utilizes Trello for most of the project planning. We have a “To Do”, “In progress”, and “Completed” section to the immediate left for immediate visibility. Going right we have individual lists for members if they desire, as well as a list of resources we are using (such as development stacks) and a list of references (such as key security points to remember).



Slack is the official avenue used for team communication. Our slack is used to talk to each other, as well as where we send updates to the Trello board.



**Build Tool/Deployment Script**

Our team has a deployment script in our Github repository. This script, when ran, will create

**Source Control**

Github (<https://github.com/illinoistech-itm/team-4-bugoverflow>) is where our code is kept as a team. We have it mostly organized, but a little more work would be needed for it to be a squeaky clean repository. Our team has had at least one commit per week, if not at least one commit per day. Our design files consist of the CSS, JS, PHP, and HTML files. These are located in the folder called “CindyUI” in the root of our repo. Documents are kept in “Documentation”. This includes information on database master-slave replication, our project papers, charts and graphs regarding how systems work together, and some other files. Deployment files are in “Deploy”. This includes a readme of instructions, a script for automated deployment, and accompanying files for the script to use to build the project. SQL files are in “Database”. Here we have “database\_schema.sql”, which builds the database and tables. The “insert.sql” file contains all of our initial data records into one file to be inserted in the database. Lastly, the main code, which is written in PHP is also kept in the “CindyUI” folder. We made this decision because the website requires the PHP files in order to properly display itself, so it made sense to group them together.

Bug tracking was originally done verbally between group members, but this quickly was proven unreliable. Then issues were tracked in Trello, in the sense that things that needed to be done or fixed were created as cards. This was greatly inefficient for actually tracking specific issues. We finally started using Github to track bugs, as we should have done from the beginning.

References

Kim, G. (2014, May 19). *DevNation 2014 - Gene Kim - Afternoon Keynote: Why Everyone Needs DevOps Now* [Video file]. Retrieved from <https://youtu.be/vTxAR8EOc8A>.

Kim, G., Humble, J., Debois, P., & Willis, J. (2016). *The Devops Handbook How to Create World-*

*Class Agility, Reliability, and Security In Technology Organizations*. Portland, OR: IT Revolution Press.

APA format (you have an errant YEAR column header) mispell Xamp (should be XAMPP) Page 1

* 1 - needed to talk about how the tools come together to represent the three ways (Gene Kim)
* 2 - Good
* 3 - Operating stack title is confusing - you mention Windows 10 in the title but you are not using Windows 10? Some people are developing on Mac, so that is in error. Also the use of XAMPP, defeats the purpose of working in Vagrant and Packer as you are to be building and deploying on the infrastrucutre you will be working on (XAMPP is convienient but you are introcuding extra dependancies and potential extra work) See WHy you need DevOps now presetnation (cut out context swithcing time of operations)
* 4 - Adequate explanation of programming languages, no citations or justifications of why those platforms were chosen5 - Good6 - Diagram was good for initial plan, but should be redone using Visio or other tooling application (see Hong's diagrams)
* 7 & 8 - Good calculations, but no information or data on how those price points were reached9 - Good use of screenshots, in your screenshot I see a task under todo that has no one assigned to it
* 10 - Good use
* 11 - Cards show signs of action, but project was demoed using XAMPP not on the Vagrant DEV platform, hence cards that are labeled done then are not actually done.
* 12 – Good
* 13 - Good use of slack for communication
* 17 - Build instructions were missing/incomplete (<https://github.com/illinoistech-itm/team-4-bugoverflow/issues/3>)
* 18 - Code submission was good, according to your punch card - most work was done in class - <https://github.com/illinoistech-itm/team-4-bugoverflow/graphs/punch-card\>
* 19-25 are missing or incompleteComments - Good overall. Needs a bit more fleshing out. Hard deatils are requried in some sections. Plans for implementing feedback loops and deployment to the DEV stage is needed. Also the last 6 items were missing or incomplete, that needs to be covered on the Trello board and resources managed to cover this oversite as well as to prioritize the automated deployment of Bugoverflow.