Full Stack Web Application Project Final Report

**Team Name: Q & A Community**

**Application Name: Q & A Community**

**Coordinator:** Kedar Ravindra Haldankar (G35856041)

**Members:**

Kedar Haldankar: G35856041: kedar16@gwu.edu

Thomas Riggs : G20988027: [Tsriggs@gwu.edu](mailto:Tsriggs@gwu.edu)

Xiran Ting: G37829558: dxr.ivory@gmail.com

Ian Davila: G38838417: [nanotane1@yahoo.com](mailto:nanotane1@yahoo.com)

Austin Moon: G20745499: abm90@gwu.edu

**Source Code Github Repository URL**: https://github.com/nanotane/QA\_project

# Application Summary

This is a Stack Overflow clone, which will provide all of the basic functionalities of Stack Overflow. The main functionalities include:

-Creating a profile

-Posting a question

-Answering a question

-Searching for questions

-Voting on questions/answers

All of the main functionality of our application is still going to be implemented. Users can create a profile. They can post a question and post an answer to a question. Users can vote on both questions and answers and the top 10 answers to a question will be displayed below on the question page. Additionally a user can search for a question based on user input. One of the functionalities that we decided not to do is tags. So, for the search functionality, it will be done based on textual analysis of the user input being cross-referenced with the database. For our secondary functionalities, we plan on implementing the rank functionality. However, we decided not to allow a user to be able to edit their own post.

# Architecture Report

*Explain all the layers and components that are part of your application.*

**Business logic:** Primarily catches errors from the DAO layer and either sends them to be handled by the next layer or handles them appropriately. Calculates the users rank, the color of the text for the user’s name, and sorting the list of answers based on rank.

Technology Report

*List ALL the technology used to build your application. Write your reasoning for choosing it.*

**Technology:** GitHub

**Use:** Version control

**Reason:** Allowed for users to work on their own time and made it easier to know what had been completed.

**Technology:** Maven

**Use:** Compiling the project with dependencies

**Reason:** Had drop wizard as a dependency and needed to compile properly with the project

**Technology:** Drop Wizard

**Use:** Allows for URL links to work with the API layer of the project

**Reason:** It is relatively easy to use and examples were readily available

**Technology:** XAMPP

**Use:** Testing and running a database

**Reason:** Easy to use and easy to set up a working database

# API Document

*Document all the RESTful APIs that are part of your application. Example:*

## Get Question

**Endpoint**: GET /questions/{questionID}

**Sample Payload**:

{“questionID” : 1111}

**Sample response**:

{“questionID”: “111”, “userID”: “222”, “qtext”: “how?”, “votes”: 10, “answers”: [{“answerID: “”112”, “userID”: “123”, “questionID”: “111”, “atext”: “this is how”, “votes”: “10”}]}

## Get Top 10

**Endpoint**: GET /questions/top10

**Sample Payload**:

n/a

**Sample response**:

{“topTenList” : [{“questionID”: “111”, “userID”: “222”, “qtext”: “how?”, “votes”: 10, “answers”: [{“answerID: “”112”, “userID”: “123”, “questionID”: “111”, “atext”: “this is how”, “votes”: “10”}]},

{“questionID”: “111”, “userID”: “222”, “qtext”: “how?”, “votes”: 10, “answers”: [{“answerID: “”112”, “userID”: “123”, “questionID”: “111”, “atext”: “this is how”, “votes”: “10”}]},

…

]}

## POST Question

**Endpoint**: POST /questions/question

**Sample Payload**:

{“userID”: “222”, “qtext”: “how?”}

**Sample response**:

{“questionID”: “111”, “userID”: “222”, “qtext”: “how?”, “votes”: 10, “answers”:[]}

## POST Answer

**Endpoint**: POST /questions/answer?={questionID}

**Sample Payload**:

{“userID”: “123”, “questionID”: “111”, “atext”: “this is how”}

**Sample response**:

{answerID: “”112”, “userID”: “123”, “questionID”: “111”, “atext”: “this is how”, “votes”: “0”}

# Development Report

*Answer the following questions in detail.*

1. What were the easiest parts of development process? And why?

Creating HTML pages and implementing some of the simple functionalities in core business logic and persistence logic.

Front-end developers have good knowledge writing HTML code. Some of the functions in back-end were very straight forward as core business logic directly calls DAO and some of the DAO functions have very simple SQL queries.

1. What were the hardest parts of development process? And how did you deal with it?

Dealing with the fact that we only had one computer with a working database program for testing. It was a real bottleneck and slowed down the development process. We dealt with it by trying to solve problems as a group.

Integrating all the individual parts was of the most difficult experience because there were unexpected server level exceptions (e.g. Server uses two different Time zones sqlException)

1. If you could make different decisions regarding any aspect of this project, what would it be and why?

We would try and integrate the project earlier so we would have more time to work out the problems between the layers. We would also install database programs on multiple computers to allow for parallel testing of the entire system.

1. How did you manage work distribution among the group members?

Front end HTML pages and javascript functions were implemented by Thomas, Ivory and Austin.

Web-APIs were implemented by Thomas

Core business logic implemented by Ian

Persistence logic and database connectivity is done by Kedar.

Finally, we all sat together to integrate all the individual parts.

1. What were the hardest parts of working in a group environment? And how did you deal with it?

As everyone in the team had totally different schedule this semester, setting up a specific time for the meeting was one of the problems. Some of the meetings were with the absence of one or two people.

Also, waiting for someone else’s part to be completed to test one’s own part was the hardest to deal with.

1. If you were to continue developing the application, what would be the next step?

We would have spent more time recognizing lower level exceptions and try to find solutions for them.

We have some extra functionalities that are already implemented in the core business logic and the persistence logic which can be used to make this web-app look more elegant and functional.

Also, to add next level of security by provisioning Facebook and Google login.