NYU TANDON SCHOOL OF ENGINEERING

Principle of Database Systems, Section B, CSGY - 6083B, Final Project - Part II, Requirement Document

March 29, 2020

1 General description

In the second part of the project, you have to create a web-based user interface for the database schema designed in the first part of project. In particular, users should be able to register, login, buy an insurance, browse and search their insurances and other content that is accessible to them. Please note that the user could be either customer or employee of WDS, and they have different authorization on data.

Note that you have more freedom in this second part of project to design your own system. You still have to follow the basic guidelines, but you can choose the actual look and feel of the site, and offer other features that you find useful. In general, design an overall neat and functional system. If you are doing the project as a group of two, note that both students have to attend the demo and know ALL details of the design. So work together with your partner, not separately. You can start by revising your design from the first part of project as needed. In general, part of the credit for this project will be given for revising and improving the design you did in the first part.

Users should be able to perform all operations via a standard web browser. This should be implemented by writing a program that is called by a web server, connects to your database, then calls appropriate stored procedures that you have defined in the database (or maybe send queries), and finally returns the results as a web page. You can implement the interface in different ways, You may use frameworks such as PHP, Java, Python or Ruby on Rails to connect to your backend database. Contact the TAs for technical questions. The main restriction is that the backend should be a relational database(MySQL recommended) based on the schema you designed in the first part, with suitable improvements as needed. The APIs of your website is recommended to be designed with RESTful style. For sensitive data such as password, it should be stored in database after

encrypted.

Your interface must take appropriate measures to guard against SQL injection and cross-site scripting attacks. To prevent SQL injection you could use stored procedures and prepare statements (if your programming language supports them). If your language does not support prepared statements, your code should check and sanitize inputs from users before concatenating them into query strings. To guard against cross-site scripting, outputs to be returned to user's browsers should be checked/sanitized to avoid scripts. Some languages provide functions, such as PHP's htmlspecialchars, to help with this. You should also define appropriate transactions to make sure that multiple users can use the site at the same time. Also, one thing to consider is how to keep state for a user session. Make sure to address these requirements(protection against SQL injections etc., and concurrency) in your write-up.

Every group is expected to demo their project to one of the TAs at the end of the semester. Since we are having our classes and all course related activities remotely for this semester term, we will figure out a way to do the demo remotely. TAs will send out notification to register for project part 2 demo and accordingly group will register for the time slots on a first come first service request basis. Both students of the group need to be present at the demo. If you use your own installation, make sure you canaccess this during the demo. One popular choice is to use a local web server, database and browser on your laptop, and share your screen with TAs during the demo. (In this case, your project can just run locally on your laptop, but you can also make your project Cloud-based). Grading will be done on the entire project based on what features are supported, how attractive and convenient the system is for users, your project description and documentation(important!), and the appropriateness of your design in terms of overall architecture and use of the RDBMS. Make sure to input some interesting data so you can give a good demo.

Describe and document your design. Log some sessions with your system. You should also be able to show and explain your source code during the demo. The documentation should consist text describing and justifying your design and the decisions you made during the implementation, and describing how a user should use your system.

There will be opportunity to get extra credit by implementing cool extra features, but extra credit is limited to about 5-10% .There may also be extra extra credit of up to 5% for doing an early demo, before the deadline. And the extra credit will be applied to project part 2 only.

2 Extra Features

Please note that the extra points are for those want to practice more and do more interesting development than basic requirements. Typical extra features could be:

1. Better design to use Cache/Containers/Serverless etc. to make your website more robust with high availability and scalability.

2. Building correct index on database to deal with the query with high frequency. In this case, you need to show us how and why you build a index, what and why this index can help to your system, and the process of your experiment/analyse to improve the performace of system by indexing. (Hint: Bottle necks of MySql, Covering index, Left-prefix index rule, Join and sort in lower-level)

3. Interesting data visulization and methods for user to interact with data,

4. Interesting features you think should be added to WDS insurance management system/website (other than basic festures of user management, insurance management)

3 Due Date of the Project

To be decided by professor

4 More Than the Project Itself

(This section won't be counted for credit)
If you want to learn more after this project, we have some interesting open-end questions for you to think about.

1. When should we use a Database and why.
2. Is NoSQL database better for this project than Relational Database and why.
3. If the user of your system increase fast (Let's say 1 million users, with total amount data of 10 TeraByte, and 1500 qps) how would you modify your architecture to make sure the service is available and data is consistent.
4. Some services require more read while some other services require more write when manipulating data. How would you choose the schema, database and architecture for different service? What if you need large-scale online data analyse.