Lab – 1 Report

**Introduction**:

Goal: The goal is to learn to develop REST services using Node.js and ReactJS, using RESTful services.

This report covers the work done for Lab-1. In Part-1, it calculator is made using Restful services, which takes two inputs from users and user can perform any of the four arithmetic operations and get the results.

In Part-2, a prototype of freelancer application is made. The server performs basic user functionality tasks like sign up a user, log in an existing user, log out and user can see his profile. Users can post their projects and can see a list of all open project’ description as well. User also has a dashboard where has the projects he bid on and the description regarding projects. This application has connection pooling with MySQL.

Report also covers the answers to the questions.

**System Design:**

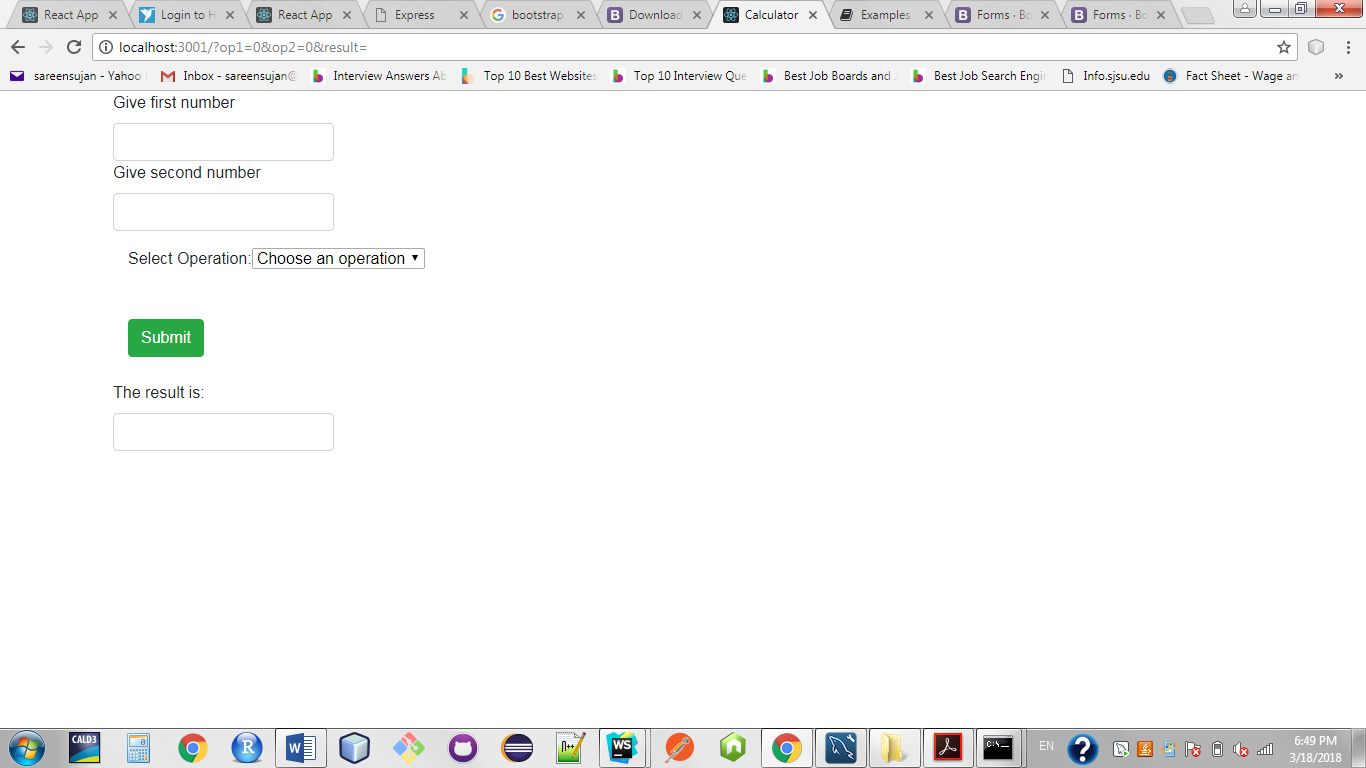
For calculator part, the UI part takes values from users and send those values to a server (using fetch API) to perform the required operation. Node.js(express) server takes the inputs, process them, and send output back to the client. And client displays output to users.

For Freelancer application, the server performs functionalities like login a user, sign up an existing user, shows up user profile, shows post project functionalities, home view, detail view, dashboard. Client is developed using HTML, ReactJS. The application has connection pooling with MySQL. Client requests to server for any required operation using fetch API. The server takes the required data from the database and performs the necessary operations and sends the output back to the client.

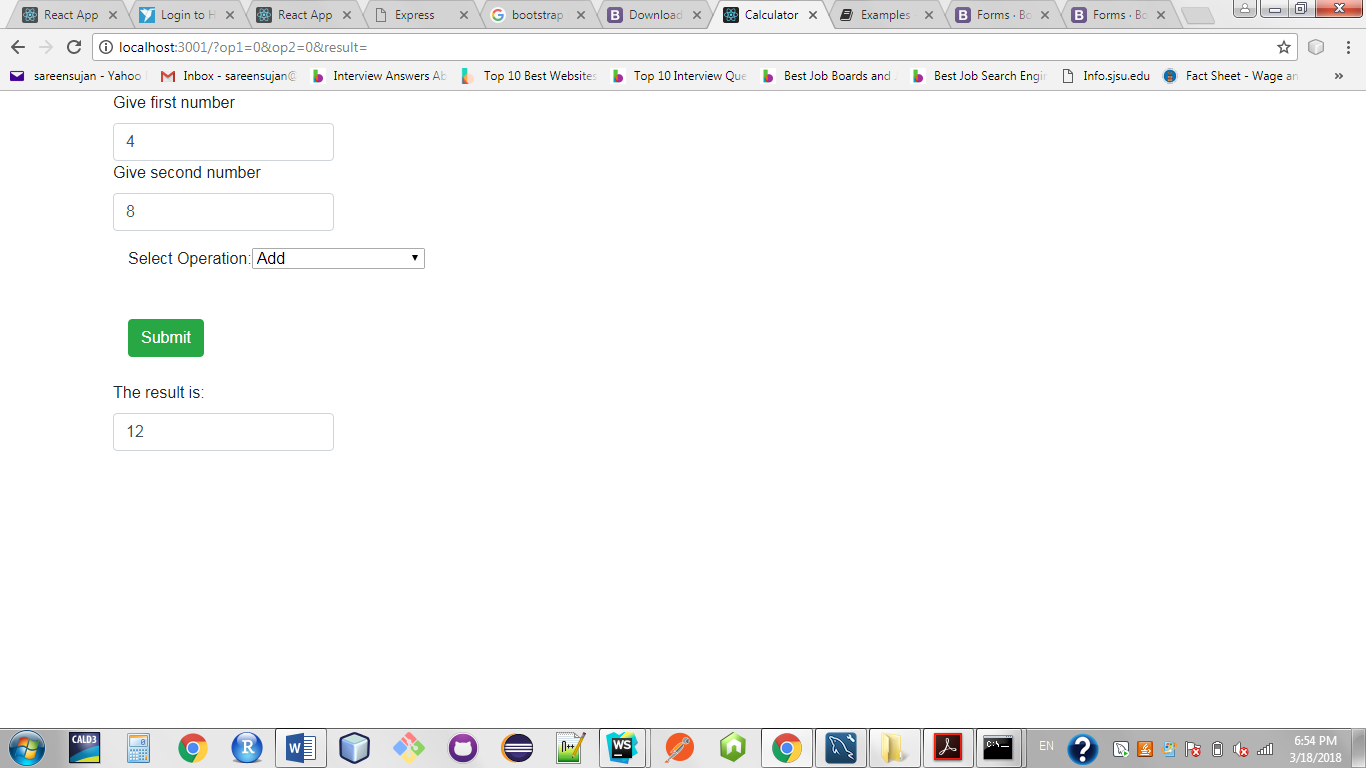
**Results**:

For Part-1:

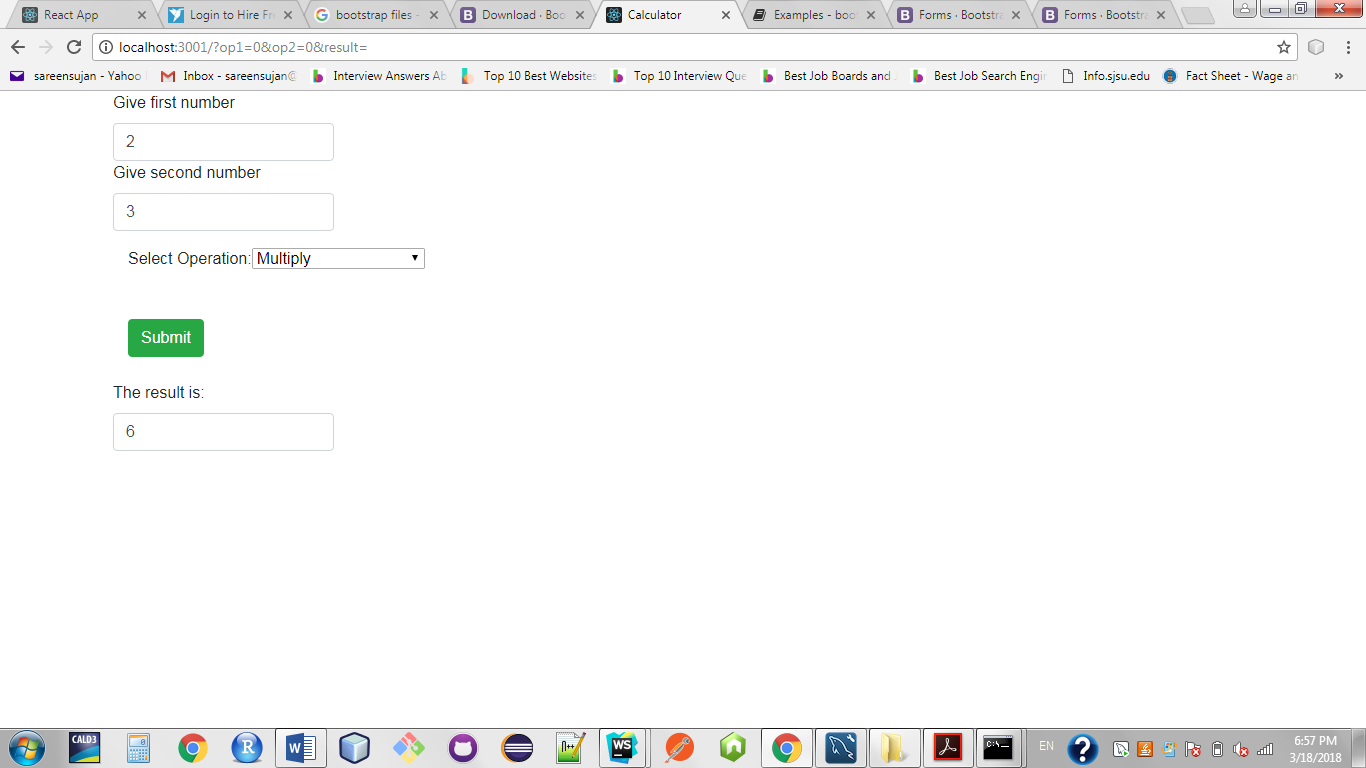
The client part of the calculator before performing any calculation:



After performing addition operation:

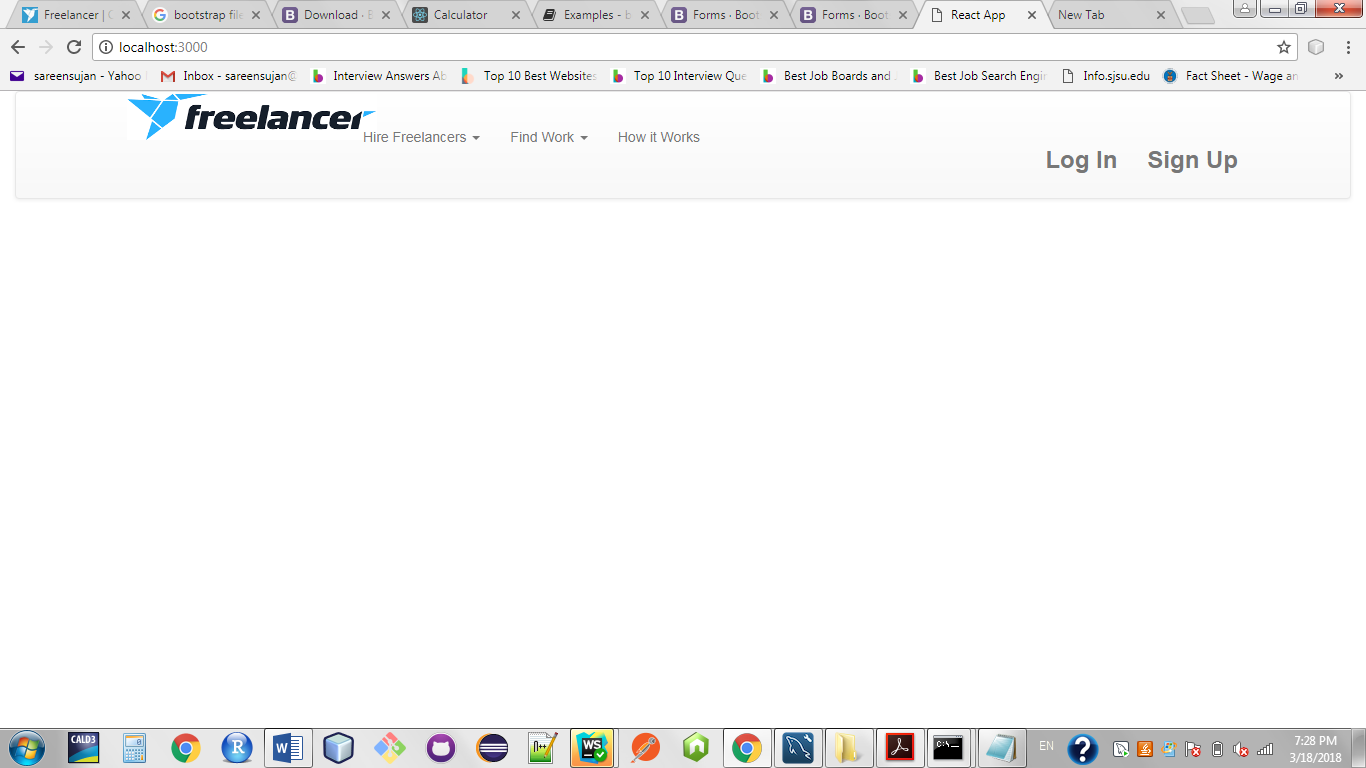


After performing multiplication operation:

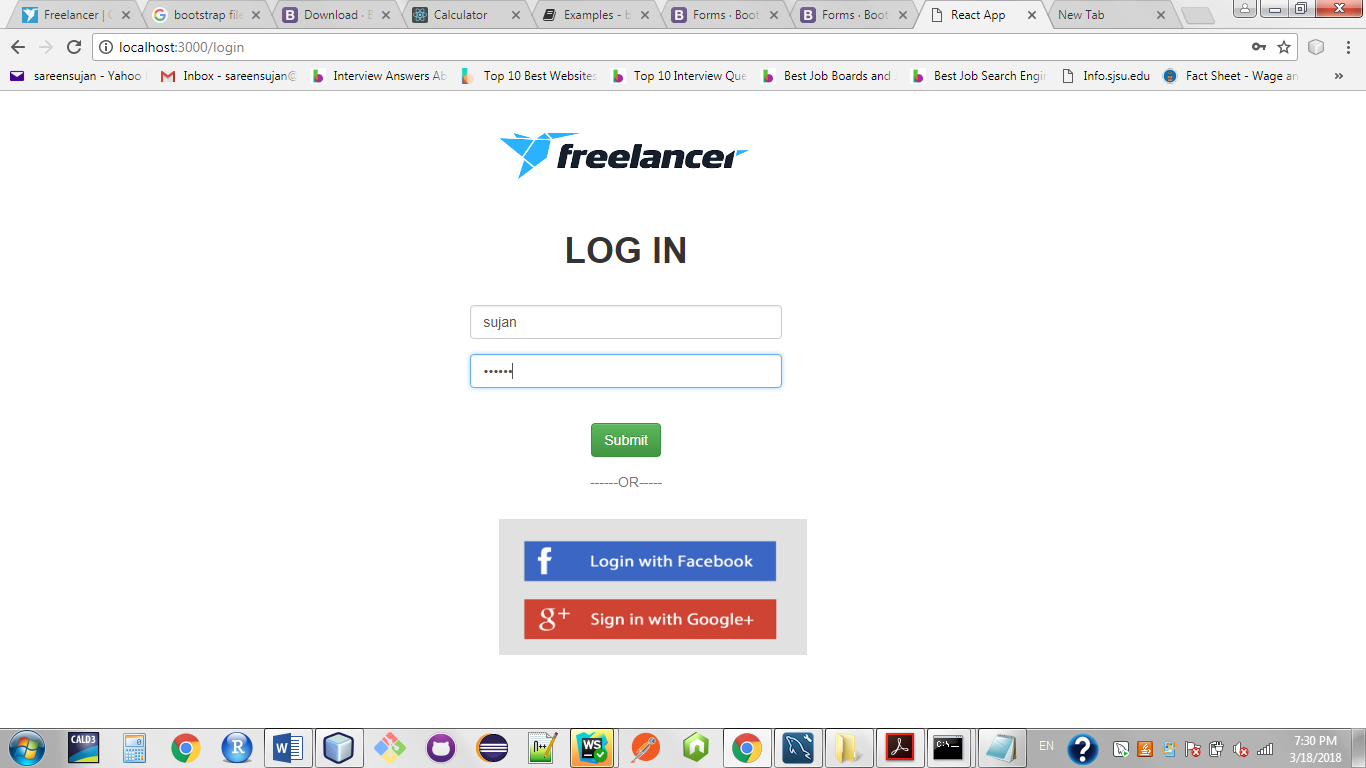


For Freelance application:

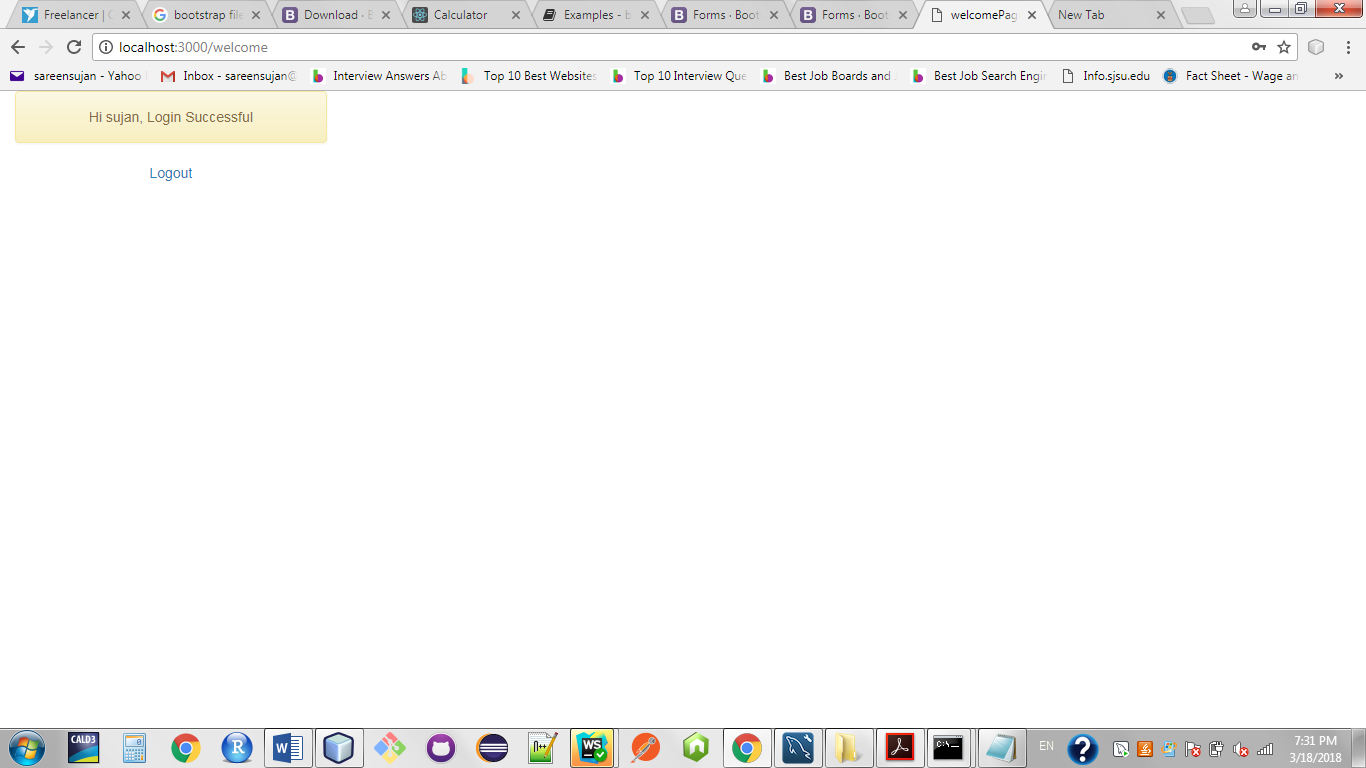
The Home Page:



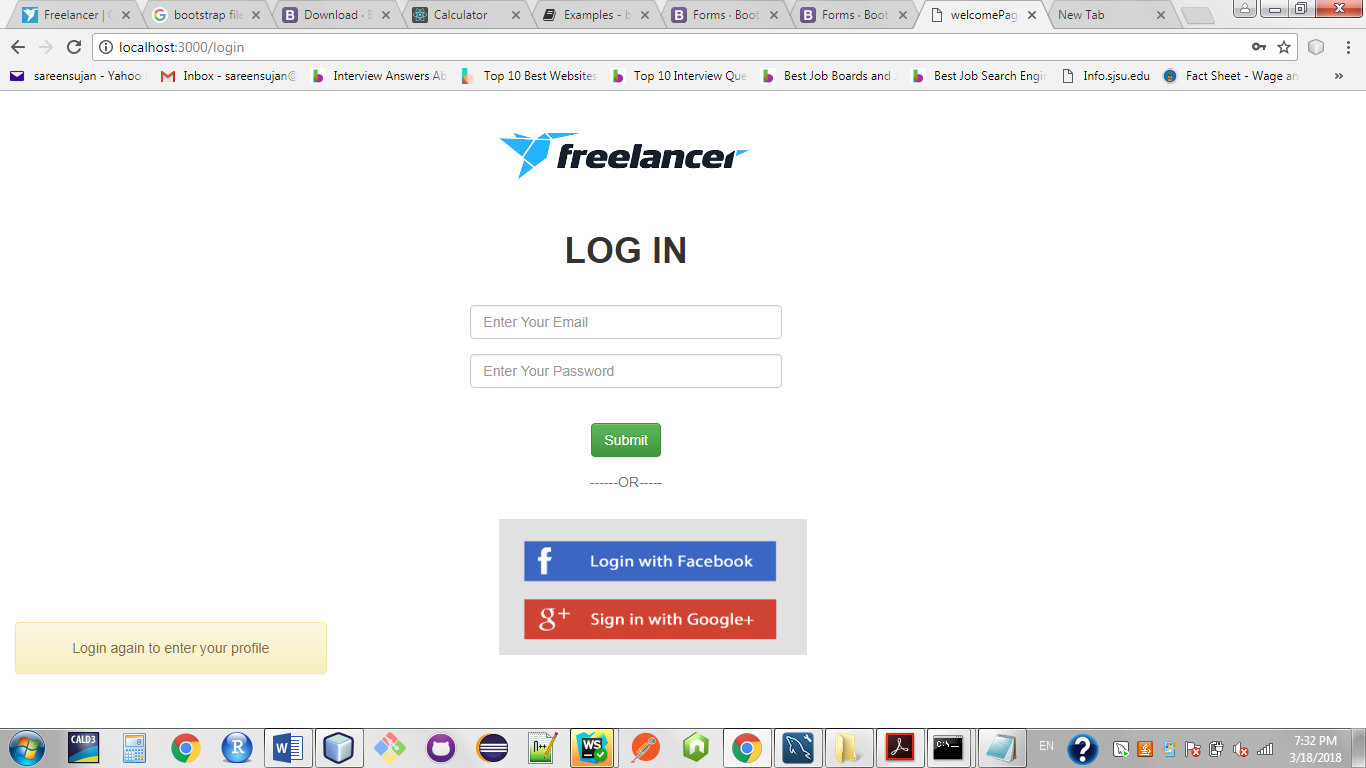
The log in page:



After user logs in:



After user logs out:



Answers to the questions:

Ans.1) Different encryption algorithms available are:

1) AES

2) RSA

3) MD5

4) SHA

I used AES (Advanced Encryption Standard) to encrypt the data and protect it against unauthorised access. It uses higher length key sizes such as 128, 192 and 256 bits for encryption. Hence, it is considered secure and effective.

Ans.2)

1. Performance: Connecting to the database is expensive and slow. Pooled connections can be left physically connected to the database, and shared amongst the various components that need database access. That way the connection cost is paid for once and spread across all the consuming components
2. Diagnostics: If there is one sub-system responsible for connecting to the database, it becomes easier to diagnose and analyze database connection usage.
3. Maintainability: if there is one sub-system responsible for handing out database connections, your code will be easier to maintain than if each component connected to the database itself.

Ans. 3) SQL caching stores the select query along with the result set. If the same query is sent again, then the server retrieves the results from the query cache. And the second time, the response time will be faster because the same query was stored previously in cache. SQL caching is shared among sessions.

Types of caching:

Procedure cache: The part of the memory pool that is used to store execution plans is referred to as the procedure cache.

Plan cache: It is a part of SQL Server's memory used to store compiled plans. It stores query plans of more than just the stored procedures.

 SQL Server buffer cache: is a place in system memory that is used for caching table and index data pages as they are modified or read from disk.

Plan caching fits better because it increases the performance.

Ans.4) Yes, session strategy I used is horizontally scalable. The session handling strategy retrieves data directly from the fast local memory and avoid access to a slow database. This helps the application in scaling linearly, just by adding more nodes to the session cluster.

GitHub credentials:

Email: [sujan.sareen@sjsu.edu](mailto:sujan.sareen@sjsu.edu)

Password: Software&1