FAULT TOLERANCE REPORT

PREPARED FOR DAYTRADING INC

PROJECT NAME:	DAY TRADING SOFTWARE
SYNOPSIS:	AN ANALYSIS OF THE FAULT TOLERANCE OF THE PROJECT; INCLUDING THE SYSTEMS REACTIONS TO FAILURE, SYSTEM AVAILABILITY, AND ERROR HANDLING.
DATE:	APRIL 7, 2015
PREPARED BY:	BRAIGHTON POLACK (V00763687), EVAN WILLEY (V00703788)

CONTENTS

1. INTRODUCTION

The purpose of this document is to outline fault tolerance of the system. The interactions of each component in the system during failure are discussed in this report, as well as the ways each component can recover from failure. The methods of error handling are also shown, which gives the user useful information in the case of any issues.

2. FAULT TOLERANCE BY COMPONENT

The fault tolerance design of each component is discussed in the sections below.

2.1. QUOTE CLIENT

The fault design of the quote client is simple, as it only communicates with two other components. In the event that the quote server goes down, it will simply continue to request a quote over and over, until it comes back up. If a transaction server goes down while requesting a quote, sending the quote response will fail, and the quote client will respond to the next request.

In the event that the quote client goes down, nothing of value is lost. The only thing lost is the cache of recent quotes, which can be retrieved again from the quote server.

2.2. DATABASE

PostgreSQL comes prepackaged with methods for fault tolerance. The database itself is designed to be fault recoverable. Even if the database is in the middle of processing a transaction and it crashes, this will not corrupt the database. Postgres is configured to wait until data is completely written to the hard drive before responding with success to the transaction server. This means commands are not buffered in memory before being written. In the event of a crash, no transactions will be lost by the database. All that is needed is to restart the server, and the Postgres database.

In addition, PostgreSQL can be configured with automatic replication to a secondary server. This allows the use of a hot standby unit, which can be switched to in the case that the main database server goes down.

2.3. LOG SERVER

The log server only receives data, so if any other component goes down it is completely unaware and unaffected. If the log server itself goes down, it will lose any logs stored in memory that have not been written to file yet. The interval that log files are written can be adjusted. Multiple log servers can be configured so that any components sending logs will simply skip to the next log server if the current one becomes unresponsive.

2.4. TRANSACTION SERVERS

The transaction servers communicate with all the other components, so the way they handle faults is the most important. If the load balancer or the database goes down while the transaction server it waiting for a response, the receive will fail. The transaction server will then attempt to reconnect to the load balancer or database until it comes back up. If the log server goes down, the transaction server is unaffected; logs will not be recorded as long as the log server is down.

Due to the fact that multiple transaction servers will be running at a time; if one goes down, the others will begin to share any active users that were sending commands to the one that is down. This is handled through a script on the web servers. All cached data will be cleared if a transaction server goes down, so when users reconnect to it, the information will need to be retrieved from the database again.

2.5. WEB SERVERS

The web servers are responsible for directing users to an available transaction server to process their commands. When a transaction server goes down while a user is connected to it, the web server will reconnect them to another available transaction server when they try to submit a new command. A script iterates through an array of known transaction servers, and selects the one with the least amount of users active on it, but immediately selects it if the user’s cached data is already present on the server. If no transaction servers are available, the command will fail, and the user will be notified that no transaction servers can be reached.

If the web server itself goes down, a user will no longer be able to reach that domain. This would require the user to manually enter a different url, associated with a separate active web server.

3. ERROR HANDLING

The error handling for the user commands is done partially on the web server, and again on the transaction server with slightly more in depth checking. In most cases an error will be caught on the web server, and the user will be immediately notified that they entered an incorrect value. The following errors are examples of this checking.

- Already have a pending buy/sell
- No pending buy/sell
- Invalid stock ID
- Invalid user ID
- Invalid price

These errors and more will also be caught on the transaction server, and it will return the following error codes to the user, along with a fail message, to describe the problem.

-10	Already have a pending buy/sell
-11	Not enough money, price is $x per stock
-12	No pending buy/sell command
-13	Quote expired
-14	Not enough money in user account
-15	Not enough stocks owned
-16	Trigger already exists for that stock
-17	Invalid stock ID
-18	Invalid user ID
-19	Invalid price
-20	Bad command

-21	Quote price is not above zero
-22	Trigger does not exist

FIGURE 3.1 – LIST OF ERROR NUMBERS

4. CONCLUSION

In conclusion, each component of the day trading system has a way of handling faults and crashes. The system is resistant to crashing in that scaled components can balance the load of transactions when one component goes down. Cached data would need to be recreated on start-up of a crashed component, and this would take some time reading information from the database. No data is lost on a system crash, as the database will not become corrupted. The database is also safely written to, as it will not respond with success until a write command is confirmed to be on the hard drive.