Assignment 1 and 2: Heartbeat Implementation.

Overview

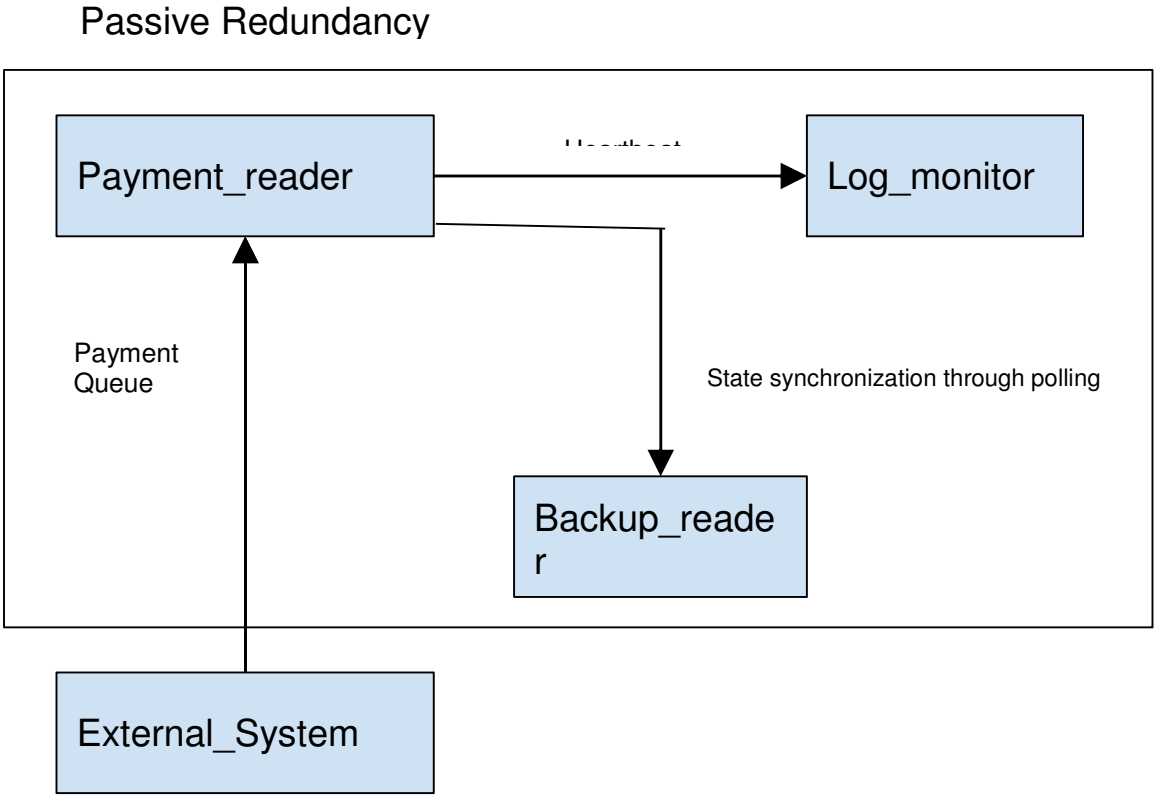

Figure 1

1. **Domain:** Banking domain
2. Overview of the critical process that is being modelled:
    - **External system:** Sends payments to the reader.
    - **Payment_reader:** This process has two threads, one thread is responsible for processing the payments and the second thread is responsible for sending heartbeats to the log_monitor
    - **Log_monitor:** This process is responsible for ensuring that the processes that it monitors are functioning correctly. This process triggers an alert to the ticketing system whenever heartbeats are skipped. The heartbeat timelines and the corresponding severity of the error messages logged are given below:

        20 seconds → Warning message is logged
        40 seconds → Major message is logged
        60 seconds → Critical message is logged

3. Non-deterministic failure in this process which makes it crash.
   In determining a non-deterministic process which stops the heartbeat, we would be setting up a flag field which would kill the thread running the heartbeat alive messages. This flag field would

be set and unset by an external process based on random function generator. The random function generator would set and unset flag based on the random values generated every 5 seconds. Further this can be extended to be based on the testing the desired availability percentage.

4. The tactics that we are planning to implement are:
   Passive redundancy - For passive redundancy, we want the system to be updated for every "x" time interval. The queue object would be passed to the other process. The payment reader module updated a queue object whenever there is a request for transaction. The queue object is static and hence multiple threads try to update it, there is a synchronisation which maintains the order of processing on First Come First Serve basis. The queue object would contain the transaction information as well as its state. After every "x" interval in the system , we would be passing the queue object which holds the information of the transaction as well as the state information, to the passive redundant system. So when the availability of the system is critical, we can still process the transactions by shifting to the redundant system.

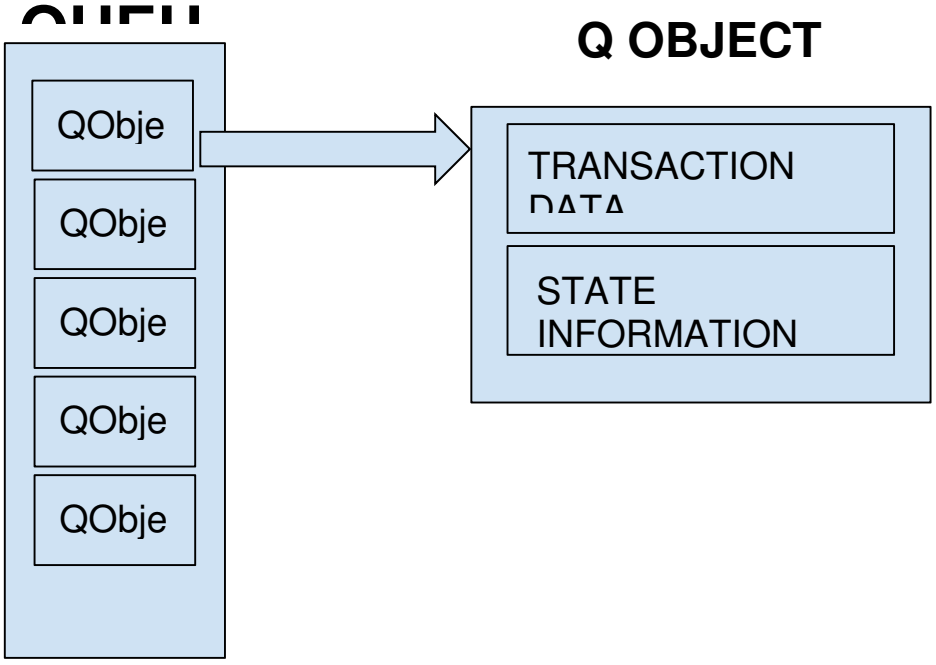

Figure 2

Implementation Snapshot

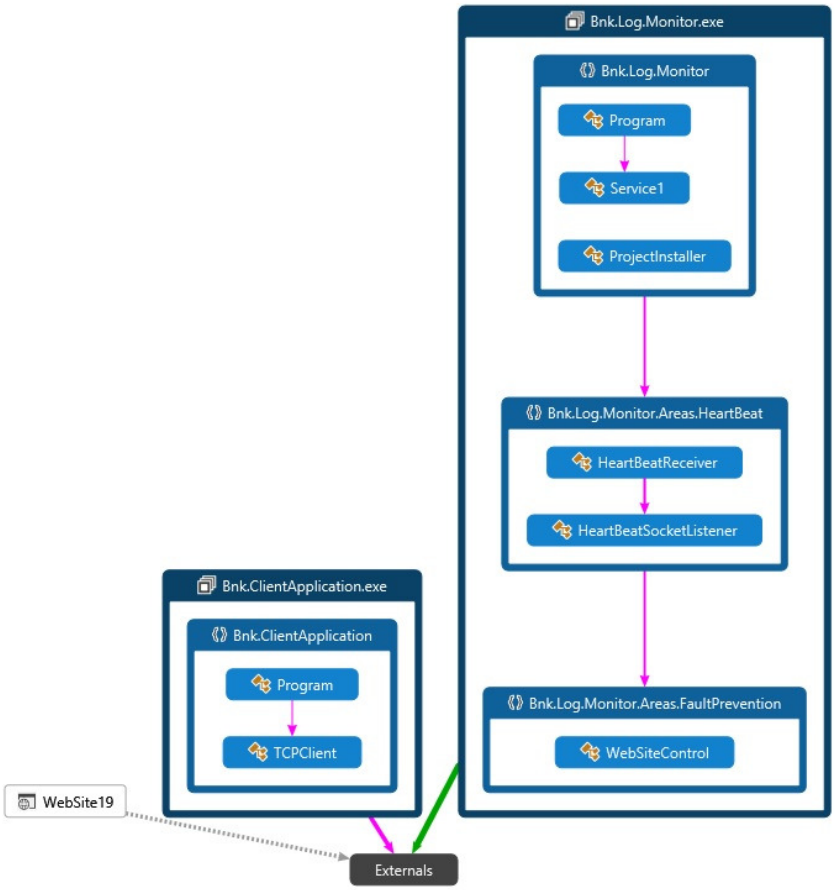

Figure 3

Technology Constraints

We are planning to implement the solution in C# using IPC and pipe objects.

Quality attribute scenario

Scenario	Source	Stimulus	Environment	Response	Response measure
Availability: When the process under consideration crashes, heartbeat	Internal	Process crash	Normal mode	The system logs a critical error and initiates the recovery	The recovery process should start and handle the processing

sender stops sending the heartbeat. The heartbeat receiver waits for pre configured time of 60 seconds. Once the 60 second mark is reached, a critical error is logged and the recovery is initiated				process	within 10 seconds.
Availability: When the process under consideration crashes, heartbeat sender stops sending the heartbeat. The heartbeat receiver waits for pre configured time of 40 seconds. Once the 40 second mark is reached, a critical error is logged and the recovery is initiated	Internal	Process crash	Normal mode	The system logs a major error and continues to wait for the process	The system should wait for the process to return to normal operation.

Readme

The below are the list of deliverables needed to setup the environment and test the implementation of heat tactic explained above:

- **Windows_Features.JPG** - Windows feature snapshot - The features that needs to be enabled are highlighted in blue box.
- **Unhandled Microsoft .NET exception.JPG** - This error has no impact, If the snapshot error is encountered, click on "No, cancel debugging"
- **Submission.zip -** This folder contains the projects

Steps:

1. Enable the features highlighted in the Windows_Features.jpg and restart the system
2. Unzip the contents of the zip file Submission.zip.
3. Solution1 contains the website, the log monitor and the test client application to test the windows service.
4. Bnk.Log.Monitor contains the log monitor and the test client application to test the windows service.
5. Bnk.Log.Monitor project
    a. Open the project and rebuild it.
    b. Open "VS2012 ARM Cross Tools Command Prompt" as an Administrator

c. Browse to Bnk.Log.Monitor\Bnk.Log.Monitor\bin\Debug and run the below command **InstallUtil Bnk.Log.Monitor.exe**
d. Run services.msc and start the "monitor Service"

6. The Target Framework for all the application and the website is 4.5. Hence you need to register 4.5 Framework in IIS.
    a. Open command prompt as an administrator
    b. Goto C:\Windows\Microsoft.NET\Framework\v4.0.30319
    c. Execute the below command
        aspnet_regiis.exe -i
7. Inetmgr
    a. Run inetmgr
    b. Right click on the sites on the left pane and select "Add Websites" and
    c. Fill the below given details:
        i. Site name - architecture11
        ii. Physical Path - C:\inetpub\wwwroot\architecture11
        iii. Port - 8050
        iv. Uncheck the "Start Website immediately" and click "Ok"
    d. Create another website by repeating Step b
    e. Fill the below given details:
        i. Site name - architecture12
        ii. Physical Path - C:\inetpub\wwwroot\architecture12
        iii. Port - 8060
        iv. Uncheck the "Start Website immediately" and click "Ok"

    Also ensure that none of the websites are running currently.
8. Solution1 project
    a. Open the project and rebuild the solution. While Building the website if there are warnings for referencing missing DLLs, then find the path to folder "Microsoft Web Tools".
        i. Usually for 64 bit Operating System - **C:\Program Files (x86)\Microsoft Web Tools**
        ii. For 32 bit Operating System - **C:\Program Files\Microsoft Web Tools**
        iii. And then correct the path of dll.refresh file displayed while double clicking on the warnings.
    b. In solution manager, right click WebSite19 and select "Publish Web App"
    c. Select architecture11 from the dropdown and click on "Publish"
    d. Repeat step b and Select architecture12 from the dropdown and click on "Publish"
9. Attach the monitor to the project by selecting keyboard"CLT+ALT+P"
    a. In the window that pops up select "Show all users" checkbox at the bottom.
    b. Select Bnk.Log.Monitor.exe process and click on Attach
    c. View in the output window of visual studio about the process log
10. Starting the website - architecture11
    a. Run inetmgr
    b. In the window that opens, click on sites to dropdown the list of websites that have been added.
    c. Rightclick on "architecture11" → Manage Website → Start
    d. Browse the website http://localhost:8050 to initiate the heartbeat messages
11. Observe in the output window mentioned in step 9 for Alive messages which occurs every 19 seconds. There is random failure invocation that stops the heartbeat from being sent. This triggers the alerting mechanism after an interval of 20 seconds. After 40 seconds it will generate a warning messages and after

60 seconds a critical alert will be logged and the Website rolls over to architecture12. This website can be accessed via http://localhost:8060. The probability of failure of heartbeat has been set at 50%.

Reference: http://www.codeproject.com/Articles/5733/A-TCP-IP-Server-written-in-C

Prerequisite:
Windows 8 or 8.1
Visual Studio 2015

Assignment 3: Performance Implementation

Quality attribute scenario

Scenario	Source	Stimulus	Environment	Response	Response measure
Performance: A customer initiates a batch payment containing 1999 transactions destined to different customers. The transactions should be processed within 2 minutes	External (User)	Initiates batch transaction	Normal Operations	~~Use thread pooling using asynchronous calls, to split the batch payment into smaller chunks and~~ Transactions are processed and status displayed	The system should process the batch payment in 2 minutes
Performance: User initiates a transaction and the transaction should get processed and the outcome displayed within 5 seconds	External (User)	Initiate a transaction	Normal operation	Transaction is processed and outcome is displayed	Transaction should be processed and outcome displayed within 5 seconds i.e. Average latency should be 5 seconds

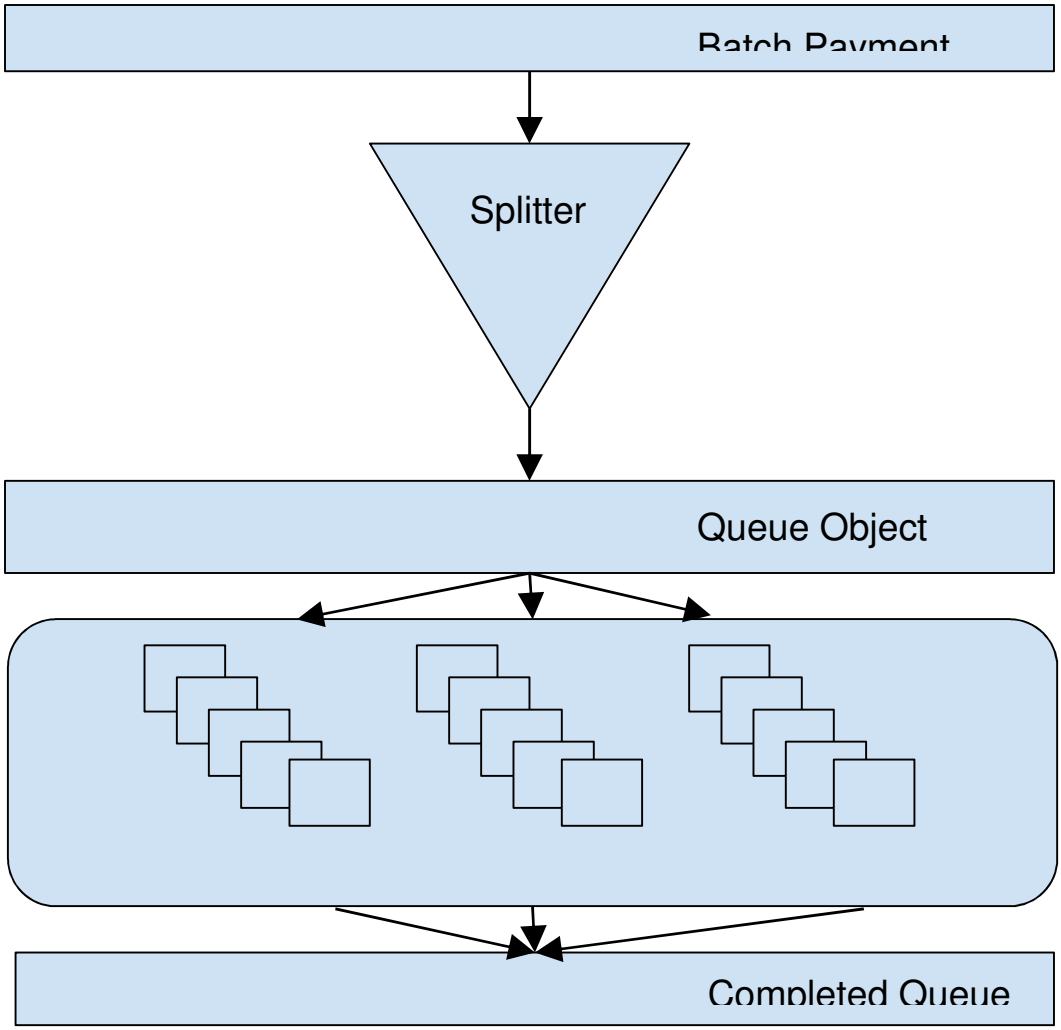

Figure 4

Batch Payment: The batch payment indicates any payments containing more than 500 transactions.
Splitter: The splitter splits any payment containing more than 500 transactions into smaller chunks with each chunk having a maximum of 500 transactions.
Queue Object: The batch payments that are split into chunks are queued for processing
Thread pool: A thread is spawn to process each chunks and once a chunk is processed the thread goes back to the queue to pick the next chunk
Completed Queue: Once the transactions are processed, they are placed in the completed queue.

Readme

The below are the list of deliverables needed to setup the environment and test the implementation of performance tactic explained above:

- **Submission.zip -** This folder contains the projects

Steps:

1. Unzip the contents of the zip file Submission.zip.
2. Solution1 contains the website and the batch file 'Batch.txt' that is to be given as input.
3. The Target Framework for all the application and the website is 4.5. Hence you need to register 4.5 Framework in IIS.
    a. Open command prompt as an administrator
    b. Goto C:\Windows\Microsoft.NET\Framework\v4.0.30319
    c. Execute the below command
        aspnet_regiis.exe -i
4. Solution1 project
    a. Open the project and rebuild the solution. While Building the website if there are warnings for referencing missing DLLs, then find the path to folder "Microsoft Web Tools".
        i. Usually for 64 bit Operating System - **C:\Program Files (x86)\Microsoft Web Tools**
        ii. For 32 bit Operating System - **C:\Program Files\Microsoft Web Tools**
        iii. And then correct the path of dll.refresh file displayed while double clicking on the warnings.
    b. Run the website from Visual Studio.
5. Click on "Transfer Amount" button on the webpage.
6. In the subsequent transfer page, upload the given batch file 'Batch.txt' and click 'Submit'
7. View in the output window of visual studio about the process log, where the threads along with hash code is displayed.
8. Clicking on "View Status" button will show the processing status of the transaction.

Prerequisite:
Windows 8 or 8.1
Visual Studio 2015

Assignment 4: Secure session management

Readme:

1. Unzip the contents of the zip file Submission.zip.
2. Open the Solution1 that contains the website in Visual Studio 2015
3. Build the solution
4. Run the Website
5. The below are the credentials for the configured users and scenario:
    a. Username: **Admin** and Password: **test1234** → Access to list of transactions performed by all account holders
    b. Username: **SMTest1** and Password: **test1234** → Access to list of transactions performed by self
    c. Username: **SMTest2** and Password: **test1234** → Access to list of transactions performed by self

    d. Username: **SMTest3** and Password: **test1234** → Access to list of transactions performed by self
6. The attached text files contain the transactions for the configured users:
    a. Batch.txt → Admin
    b. Batch11.txt → SMTest1
    c. Batch12.txt → SMTest2
    d. Batch13.txt → SMTest3
7. Login in as any of the configured user and the below can be observed in the output window of Visual Studio(**Authentication Scenario**):
    a. Associated Session ID
    b. GUID of the logged in User
    c. The Role of the User whether **'Administrator'** or **'AccHolder'**
8. Click on 'Transfer Amount', it navigates to Transfer page. Select the batch file depending on the user that has logged in.(Batch files associated with each user is mentioned in point 6) and click 'Submit'
9. Based on authorization and configured roles, if the user is 'Admin' a 'View All Transactions' button is displayed. If not 'View Transactions' button is displayed. Clicking on the displayed button, will give the transactions based on the criteria explained in the previous point.(**Authorization Scenario**)
10. When no users are logged in , clicking the 'Amount Transfer' button will take us to the login page(**Authentication Scenario- Denying access to non authenticated users**)

Prerequisite:

Windows 8 or 8.1
Visual Studio 2015

Assignment 5: Architecture Breakers - Secure session management

Architecture Breakers List

Breaker ID	Breaker	Description
BR-01	User shall not be able to access "Transfer Amount" functionality without logging in.	All users who needs to use the "Transfer Amount" functionality/task should be properly authenticated.
BR-02	All users except admin shall be	All users except admin shall be able to access only the details of the transactions executed by them and not the details of the transactions

	able to access only the details of the transactions executed by them.	executed by other users.
BR-03	Each user shall be identified by a unique session id	A unique session needs to be generated and associated with each authenticated user.
BR-04	A user shall be able to execute transactions pertaining to only their accounts	An authenticated user shall be able to execute the transactions only on his account and not on any other accounts

Test Case

Test Case	Breaker-ID	Test Scenario	Test Steps
TC - 01	BR-03	Successful Test	1. Open BnkApplication Project 2. On Solutions Explorer on the Project BnkApplication.Tests, Open the file in TransferDetailTest.cs in Old_App_Code Folder 3. Right click in the file and click Run tests 4. Both the test would be executed in unique session id test would pass
TC - 02	BR-02	Failed Test	1. Open BnkApplication Project 2. On Solutions Explorer on the Project BnkApplication.Tests, Open the file in TransferDetailTest.cs in Old_App_Code Folder 3. Right click in the file and click Run tests 4. The test would be executed and transfer detail test case would fail.

Prequisite:

1. Nuget Package: Moq, Entity Framework
2. Visual Studio 2015
3. Windows 8 or 7
4. In TransferDetailServiceTests as well as LoginServiceTest file of BnkApplication.Test, Database mdf needs to be pointed to relative location of the file in the machine where the project is downloaded.

P.S:Both the test would be executed in one go , unique session id test test case would pass and transfer detail test case would fail.