Prime Time

Software Requirements Document

ECE 480

October 16, 2016

Ibrahim A, Collin Rokke, Tarin Sultan, Mike Maryn, Vehans Ayvazi

Table of Contents

1. Introduction

1.1. Purpose

The main objective of this project is to utilize concepts from the article, *Real-Time Maintenance Prioritization with Learning Capability* by Dr. Meng-Lai Yin and Andrew J. Cha, to solve a problem. In particular, the utilization of two distinct thought processes shall both solve problems as well as recall their solutions. With this project, the specific problem to be solved shall be the identification of prime numbers.

1.2. Scope

This software shall implement a number of ways to find prime numbers, ideally in a more efficient way. The software shall use a graphical interface to interact with the user and get their input. A number entered shall undergo the prime determination algorithm which utilizes the slow brain thought process and sends back determination times as well as an indicator for whether or not the number is prime. The second time that number is entered, the software shall skip the determination algorithm and go straight to the database pulling out that number from the memory and sending back much faster determination times. Future implementations shall include hopefully a more agile determination using probability analysis.

1.3. Glossary

Fast Brain (FB)	Recalling memories (Database Method)
Slow Brain (SB)	Solving New Problems (Algorithm Method)
GUI	Graphical User Interface (Note: UI = User Interface)
Flush	Total deletion of everything saved thus far
Admin	Development and Test team
Client(ele)	Any user(s) of the system, uninvolved in development; Reports errors to Admin
POC	Proof of Concept
DB	Database
SW	Software
SNG	Sequential Number Generator

1.4. References

[1] *Real-Time Maintenance Prioritization with Learning Capability* by Dr. Mengali Yin, and Andrew J. Cha.

2. Overall Description

2.1 Product Perspective

This project shall provide a simple interface with which a user can determine whether or not their input number is prime. The software shall implement both fast and slow brain thought process to provide the user with quick and accurate results. This software shall utilize the slow brain thought process by allowing the number entered to undergo a prime determination algorithm. In addition to that, this software shall utilize the fast brain thought process allowing a search to be conducted for the number in our generated database of primes. via the database. Ideally, a third thought process shall be implemented in which probability data shall be utilized to provide even faster results.The software shall be able to decrease the time needed to find prime numbers by using the algorithm, the memory as well as the analytics.

2.1.1. System Interfaces

The interface for this system shall be a simple GUI. With a complete and functional base interface.

2.1.2. User Interfaces

The interface with the user shall be a simple, software-based GUI. The user shall enter an integer with expectations that the software shall output whether that integer is prime or not.

2.1.3. Memory Constraints

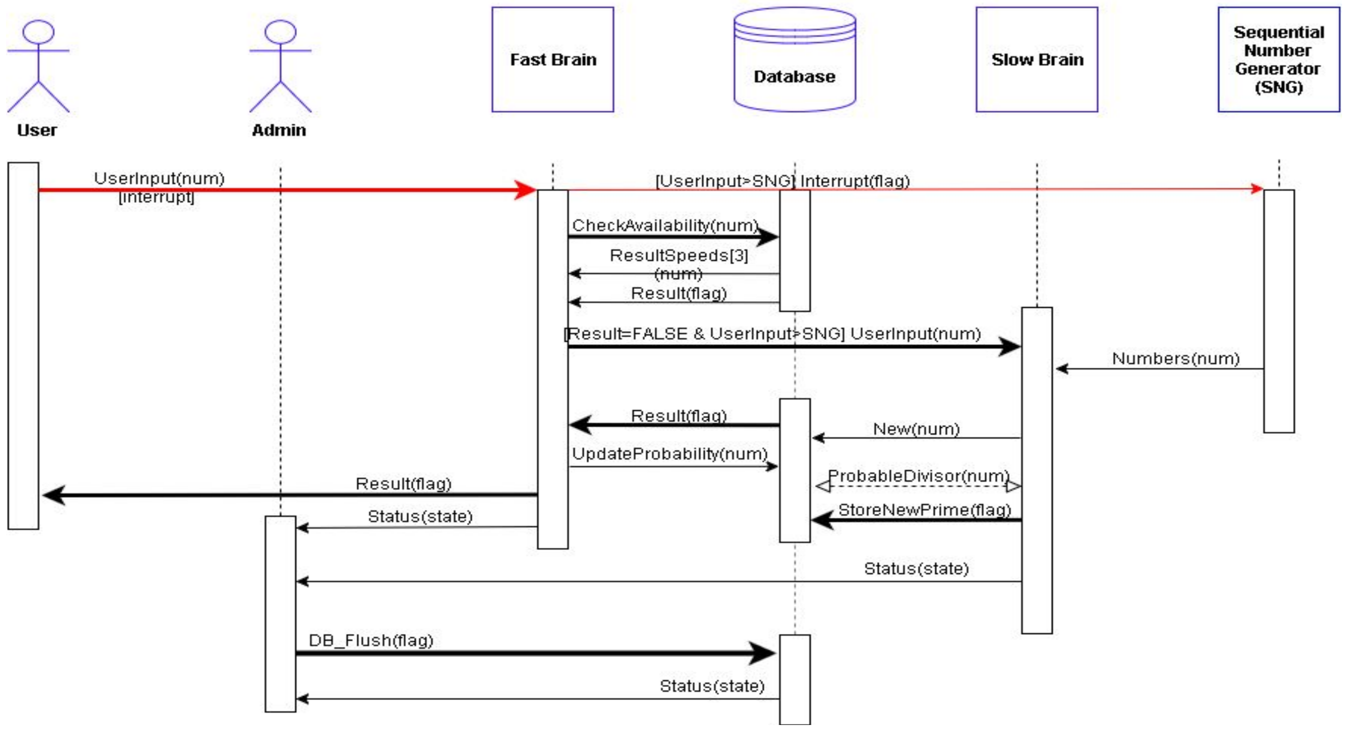
Memory constraints shall be entirely dependant upon the space allocated for the database. The sequential generator/slow brain algorithm shall quickly fill a smaller database. Much consideration should be given on the system in which this software is to be installed.

2.1.4. Operations

This software shall be in operation from when the program is executed until the program is exited. Should the maximum memory be reached, the database shall be force-flushed and normal operation shall begin again.

2.2. Product Functions

Sequence diagram:



2.3. User Characteristics

The projected users of this software are the admins and the clients.

2.3.1. Admin Characteristics

Admins shall have indicators to be able to verify which of the systems are operational. An admin shall have complete access, including a command to flush the entire database. This role shall grant a clearance of all security permissions user.

2.3.2. Client Characteristics

Clients shall strictly have access to the GUI. Here, they shall have the ability to input a number and see output for if it is a prime number or not as well as a fastest calculation and/or seek time for that number.

2.4. Constraints

This system shall have the ability to handle one user at time using the single software-based GUI on the demonstration machine.

3. System Requirements

3.1. External Interface

3.1.1. Admin Interface

Admins shall have complete access to all system code, GUI interfaces, and database commands to manage the overall system by performing maintenance and verifying that the system is operational. Most of an admin’s interaction with the system shall be via code for development and testing.

3.1.2. Clientele Software Interface

Clients shall strictly have access to the GUI interface where they shall enter an integer via keyboard to have the software determine if their integer is prime or not. They shall be able to see seek/calculation times and an indicator for prime results, but shall not be able to modify any parameters.

3.2. Functions

3.2.1. Status/Testing Function

Each software component shall be able to display its operational status to an admin for troubleshooting and verification purposes. The admin shall have a flush command for testing and maintenance purposes.

See Figure 6.3 for testing state transitions.

3.2.2. GUI Function

The software shall get user input from the GUI and check the input against the sequential number generator and send the input to the proper Brain. If the user input is not stored in the database then it shall be sent to the slow brain which in this case is the prime determination algorithm. However if the user input has already been entered before, then then it shall be sent to the fast brain which is the database. Once accurate results are returned, the GUI shall display a multitude of information about the input number including whether the user input is prime or not as well as the determination times for both the slow brain and fast brain. If no user input is received, a random number generator shall act in place of the user.

3.2.3. Database Storage Function

Once data is received, the software shall determine which of it is to be stored with the appropriate associated prime and which of it can be disregarded/discarded.

3.2.4. Database Sorting Function

The data collected shall need to be sorted in various ways. This function, when called upon, shall organize our data in ways that shall be beneficial to the performance of the prime number validation.

3.2.5. Fast Brain Function

Once input is received, the fast brain shall search for it in the database and return the results to the GUI. The seek time shall be checked against the database and updated if necessary.

3.2.6. Slow Brain Function

Once input is received, the slow brain shall attempt division with each number less than the input. Every integer divisor and indicators shall be passed to the database storage function and back to the GUI.

3.2.7. Probability Analysis Function

With Probability data collecting for every number passing through the sequential generator, we hope to be able to utilize the data for a more effective slow brain algorithm. With a forever expanding database, searching more probable prime divisors first shall hopefully yield much faster results.

3.3. Performance Requirements

3.3.1. System Performance

User input sends an interrupt flag thus making the user input take priority in the entire system.

3.3.2. Search Performance

With the collection of probability data, an optimized search algorithm can be put into operation which shall significantly cut down on slow brain calculation times for sequential numbers. This shall require the database to be optimally sortable to quickly and accurately find the potential divisors with the highest probability.

3.3.3. Monitoring Performance

All performance ratings shall be communicated to the Admin through the status function(s).

3.3.4. Database Performance

The database should be appropriately scalable and operate on a sufficient system so as not to run out of storage space rapidly.

3.4. Logical Database Requirements

3.4.1 Database Population

The program shall be able to populate the database with information each time a new value is generated.

3.4.2 Stored Information

Database shall contain number generated, calculation time, seek time, and a probability of being a divisor in a non-prime number.

3.5. Design Constraints

This Software shall comply with “good” software coding standards. Each engineer in employ has this built into their contract.

3.5.1. Efficiency Compliance

Optimization of the software to make efficient use of time and resources of the system which shall be under test.

3.5.2. Maintainability Compliance

The engineers shall properly comment their code and collaborate on github for complete input of all members. Functions within the code should be readable.

3.6. Software System Attributes

3.6.1. Reliability

The software shall be functional during execution.

3.6.2. Availability

The software shall be accessible by the users and admins only.

3.6.3. Security

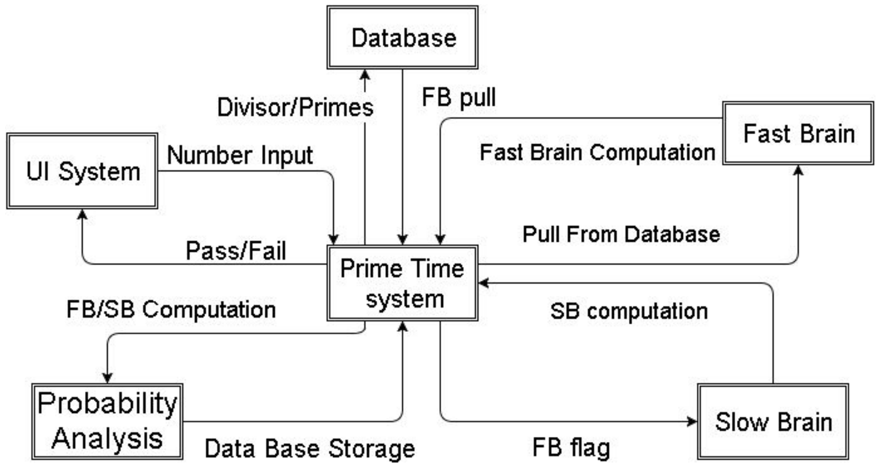
The software shall be secured enough to protect itself from hacker, users who can damage the system, etc.

3.6.4. Maintainability

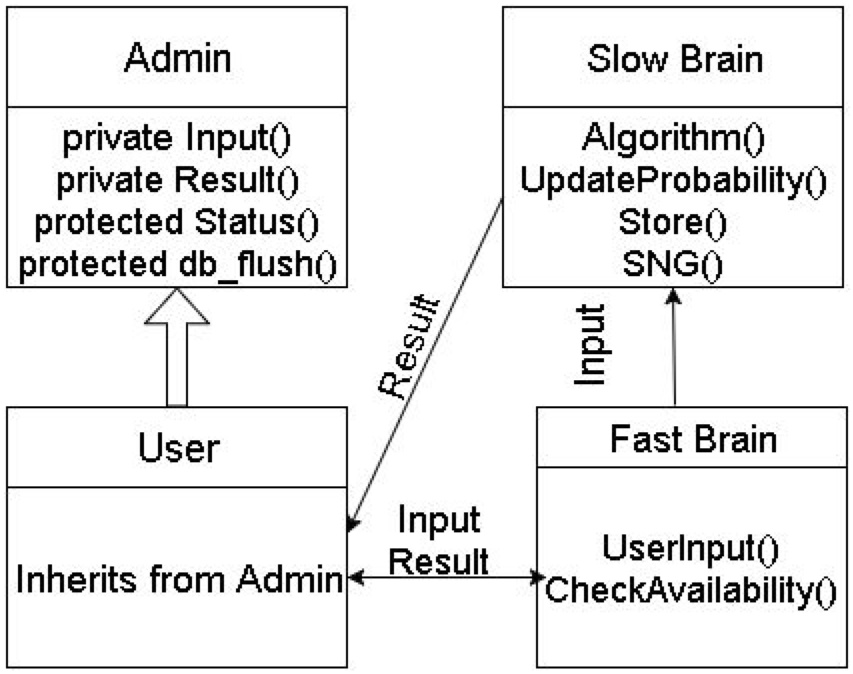
The code shall be well documented and most modules shall be reusable.

4. System Models

4.1. Business Process Model



4.2. Class Association



5. System Evolution

5.1. Software Development

System design with software based visual representation and maintenance solution design.

5.2. Hardware Integration

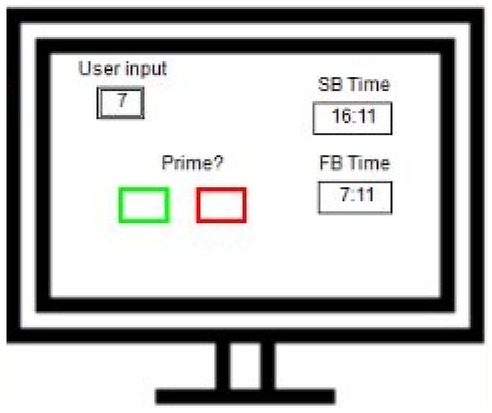
Further system design with hardware based visual representation.

5.3. User Interaction Development

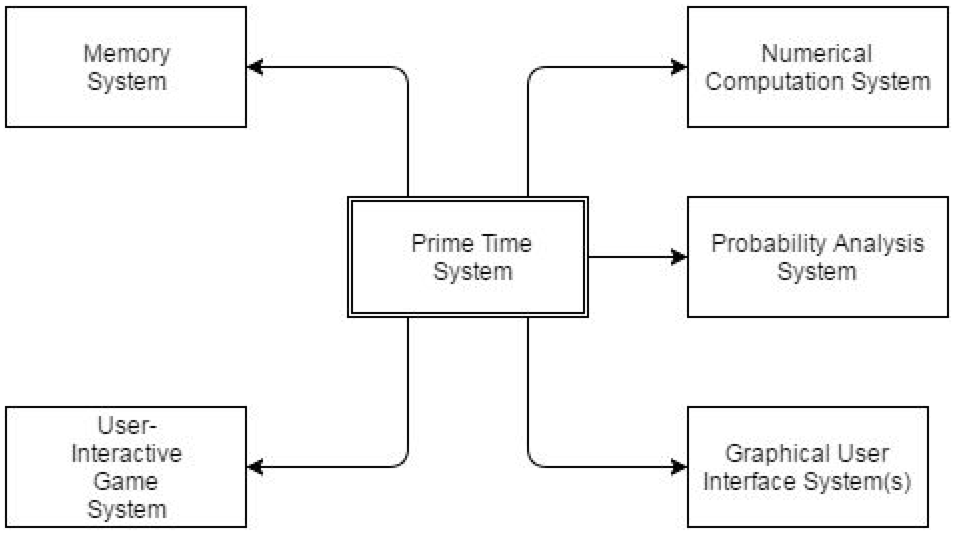
See Appendix 6.1 diagram for UI interaction.

6. Appendices

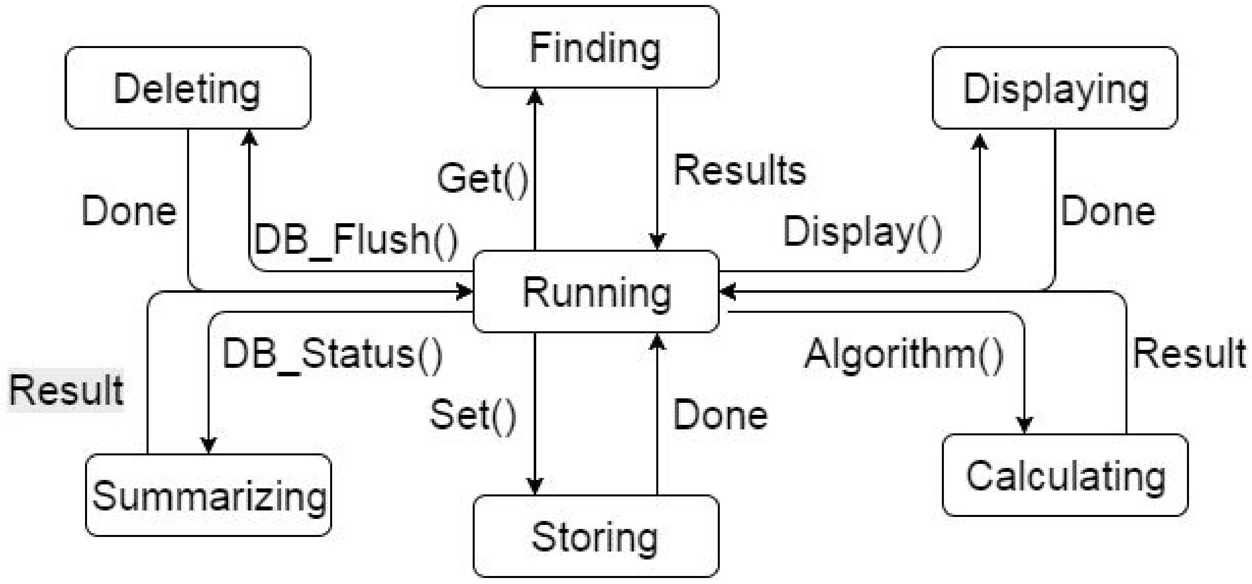
6.1 User Interaction Diagram



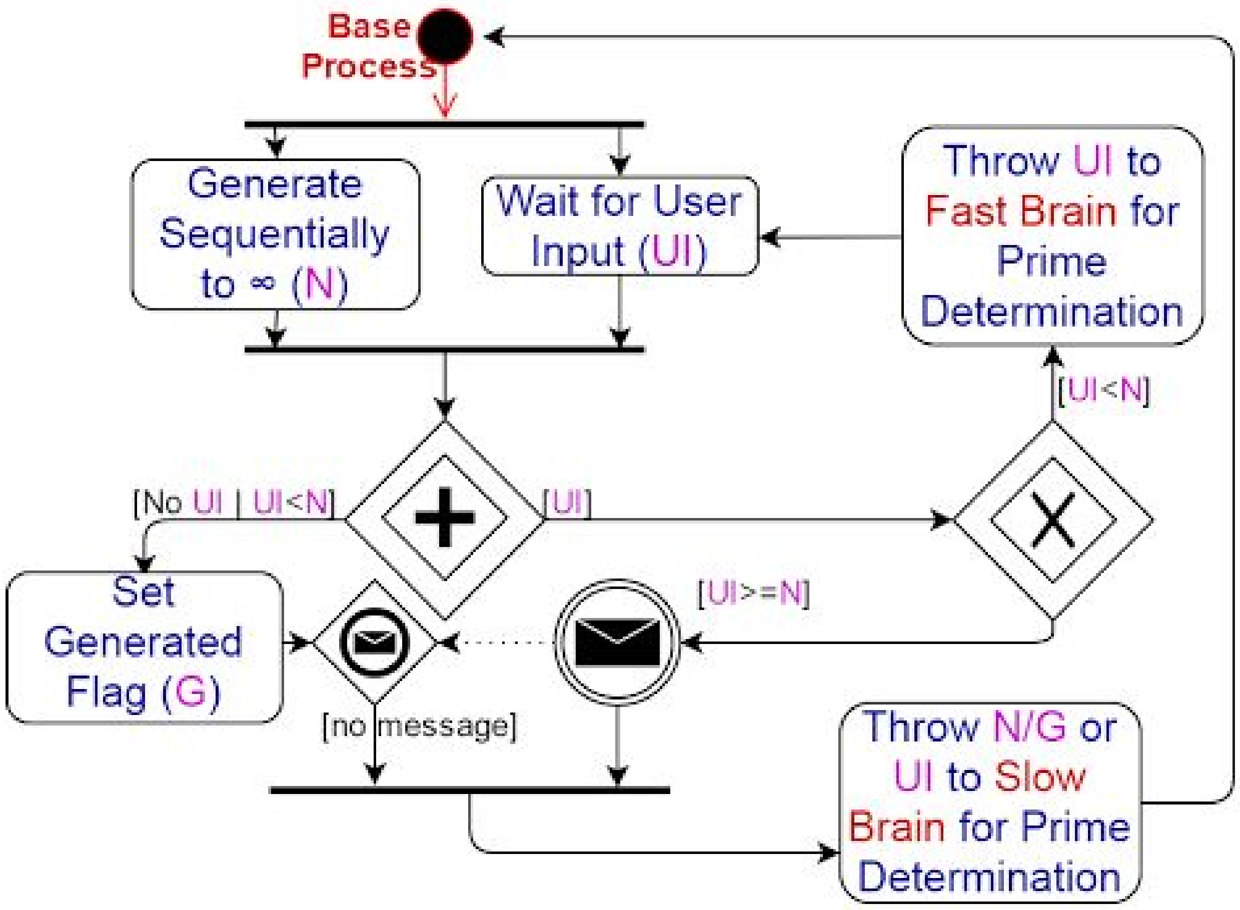
6.2 Context Model

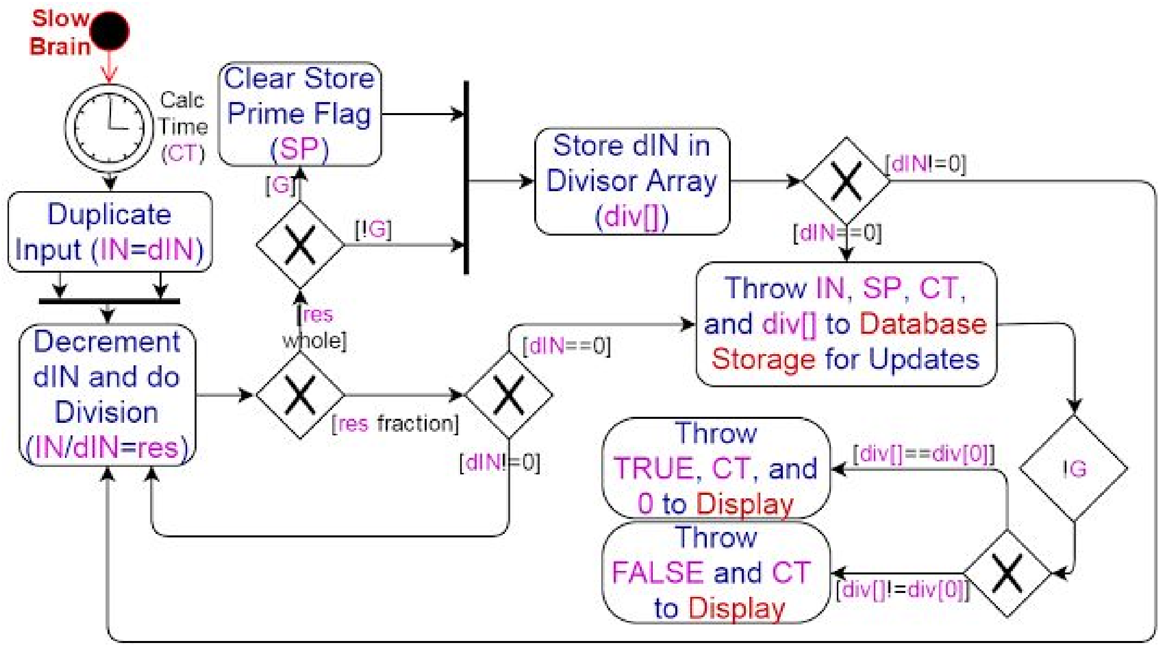

Memory System
Numerical Computation System
Prime Time System
Probability Analysis System
User-Interactive Game System
Graphical User Interface System(s)

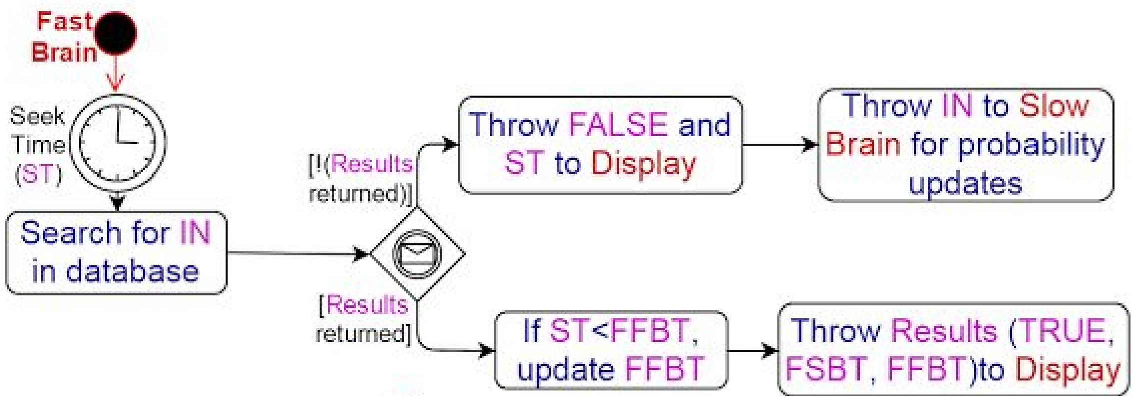
6.3 Testing State Transitions Diagram

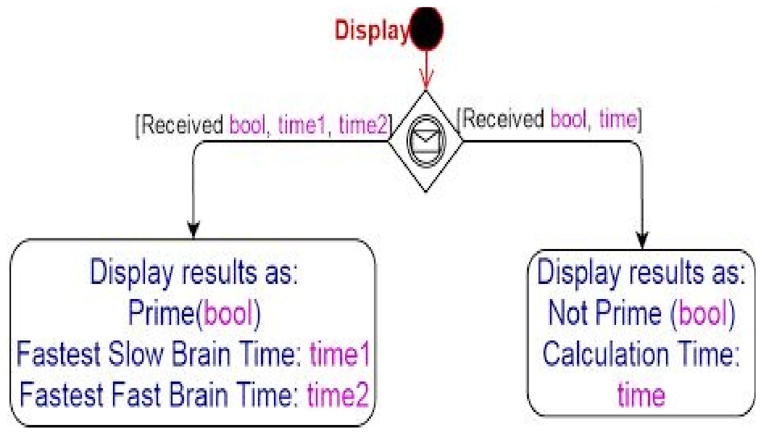


6.4 Activity Diagram


Base Process
Generate Sequentially to ∞ (N)
Wait for User Input (UI)
Throw UI to Fast Brain for Prime Determination
[UI<N]
[No UI | UI<N]
[UI]
Set Generated Flag (G)
[UI>=N]
[no message]
Throw N/G or UI to Slow Brain for Prime Determination


Slow Brain
Calc Time (CT)
Duplicate Input (IN=dIN)
Clear Store Prime Flag (SP)
[G]
[!G]
Store dIN in Divisor Array (div[])
[dIN!=0]
[dIN==0]
Decrement dIN and do Division (IN/dIN=res)
[res whole]
[res fraction]
[dIN==0]
[dIN!=0]
Throw IN, SP, CT, and div[] to Database Storage for Updates
!G
Throw TRUE, CT, and 0 to Display
[div[]==div[0]]
Throw FALSE and CT to Display
[div[]!=div[0]]


Fast Brain
Seek Time (ST)
Search for IN in database
[!(Results returned)]
Throw FALSE and ST to Display
Throw IN to Slow Brain for probability updates
[Results returned]
If ST<FFBT, update FFBT
Throw Results (TRUE, FSBT, FFBT)to Display


Display
[Received bool, time1, time2]
[Received bool, time]
Display results as: Prime(bool) Fastest Slow Brain Time: time1 Fastest Fast Brain Time: time2
Display results as: Not Prime (bool) Calculation Time: time