**INTELL**

Software Design Document

Name (s): Hamza Iqbal,

Supervisor:

Dr. Hammad Afzal

Dr. M. Zaki Murtaza

M. Umar Farooq,

Muneeb Ullah Mashhood

Bese-16

Date: (Feb/12/2014)

**TABLE OF CONTENTS**

[1. INTRODUCTION 3](#_Toc380060578)

[1.1 Purpose 3](#_Toc380060579)

[1.2 Scope 3](#_Toc380060580)

[1.3 Overview 4](#_Toc380060581)

[1.4 Reference Material 4](#_Toc380060582)

[1.5 Definitions and Acronyms 4](#_Toc380060583)

[2. SYSTEM OVERVIEW 4](#_Toc380060584)

[3. SYSTEM ARCHITECTURE 5](#_Toc380060585)

[3.1 Architectural Design 5](#_Toc380060586)

[3.2 Decomposition Description 6](#_Toc380060587)

[3.2.1 Data flow Diagrams 6](#_Toc380060588)

[3.2.2 Activity Diagram 7](#_Toc380060589)

[3.3 Design Rationale 30](#_Toc380060590)

[3.3.1 Bridge php with c++ 30](#_Toc380060591)

[3.3.2 Bridge php with java 30](#_Toc380060592)

[4. DATA DESIGN 30](#_Toc380060593)

[4.1 Data Description 30](#_Toc380060594)

[4.2 Data Dictionary 32](#_Toc380060595)

[4.2.2 JavaServer Module 32](#_Toc380060596)

[5. COMPONENT DESIGN 33](#_Toc380060597)

[5.1 PHP-Controller 33](#_Toc380060598)

[5.1.1 Module Description 33](#_Toc380060599)

[5.1.2 Module Specification 33](#_Toc380060600)

[5.2 PHP-Model 35](#_Toc380060601)

[5.2.1 Module Description 35](#_Toc380060602)

[5.2.2 Module Specification 36](#_Toc380060603)

[5.3 PHP-View 36](#_Toc380060604)

[5.3.1 Module Description 36](#_Toc380060605)

[5.3.2 Module Specification 36](#_Toc380060606)

[5.4 Java Communication module 36](#_Toc380060607)

[5.4.1 Module Description: 36](#_Toc380060608)

[5.4.2 Module Specification: 36](#_Toc380060609)

[5.5 JavaServer Module 37](#_Toc380060610)

[5.5.1 Module Description: 37](#_Toc380060611)

[5.5.2 Module Specification: 37](#_Toc380060612)

[a) Register execution module 37](#_Toc380060613)

[b) SendJobRequest 38](#_Toc380060614)

[c) SendResultToController 38](#_Toc380060615)

[5.6 JavaExecution Module 39](#_Toc380060616)

[5.6.1 Module Description 39](#_Toc380060617)

[5.6.2 Module Specification: 39](#_Toc380060618)

[ii. Generate xml 39](#_Toc380060619)

[iii. SendResultToJavaServer(String xmlpath) 39](#_Toc380060620)

[6. HUMAN INTERFACE DESIGN 40](#_Toc380060621)

[6.1 Overview of User Interface 40](#_Toc380060622)

[6.2 Screen Images 41](#_Toc380060623)

[6.3 Screen Objects and Actions 45](#_Toc380060624)

[7. REQUIREMENTS MATRIX 45](#_Toc380060625)

[8. APPENDICES 46](#_Toc380060626)

# INTRODUCTION

## Purpose

INTELL stands for INTEgrated tool to paraLLelize and evaluate serial code. It is part of a project that is intended for the automated conversion of serial code to the optimized parallel code. INTELL utilizes an underlying polyhedral loop transformation and code generation framework called CUDA-CHiLL, which takes Transformation Strategy Generator (TSG) generated strategies as input and produces CUDA code.INTELL also represents graphical comparison between serial code and the optimized parallel code.

## Scope

The final product shall enable the users to utilize the GPU on the server for research purposes by allowing them to test generated code of commonly used benchmarks on specified GPUs without obtaining the expensive hardware. This will provide the users the ability to conduct their research through a Web Interface thus providing portability.

The product will also provide a test bed to the users which will allow them to compare the parallelization of code and give them a measure of optimization through parallelization. The product will utilize the CUDA-CHILL Framework for automated production of parallelized CUDA-Code from serial ‘C’ code, tailored for the GPU on the server.

The goals it shall achieve are

* It will provide a Graphical User Interface for CUDA-CHILL.
* It will provide a test bed for comparing the code.
* This middleware will act as a mediator between user and the backend framework on the server.
* The interface will be Web Based Interface thus providing increased portability to the user.

Main advantages that could be achieved from this system are

* Comprehensive test bed.
* Easy parallelization of serial code.
* Promote access to services through Web Interface.

## Overview

This design document is a written description of INTELL. This will give our team overall guidance to the architecture and design of the software project. This document contains architecture diagrams with pointers to detailed feature specification of smaller pieces of design. This design document will be a stable reference, outlining all parts of the software and how they will work. The document gives a detailed description of INTELL, while maintaining a high-level view of the software.

## Reference Material

* Autotuning, Code Generation And Optimizing Compiler Technology for GPUs by Malik Muhammad ZakiMurtaza Khan.
* A Script-Based Autotuning Compiler System to Generate High-Performance CUDA Code by
  + Malik Khan, USC/ISI; Marina del Rey CA; National University of Science and Technology, ISB, Pakistan
  + ProtonuBasu, University of Utah; Salt Lake City, UT
  + Gabe Rudy, University of Utah; Salt Lake City, UT
  + Mary Hall, University of Utah; Salt Lake City, UT
  + Chun Chen, University of Utah; Salt Lake City, UT
  + Jacqueline Chame, USC/Information Sciences Institute; Marina del Rey CA
* ISO 9241-11:1998
  + Ergonomic requirements for office work with visual display terminals (VDTs) -- Part 11: Guidance on usability

## Definitions and Acronyms

Hence forth this document refers to the Software Design Specification of INTELL which stands for INTEgrated tool to paraLLelize and evaluate serial code. Hence for in this document this system refers to INTELL.

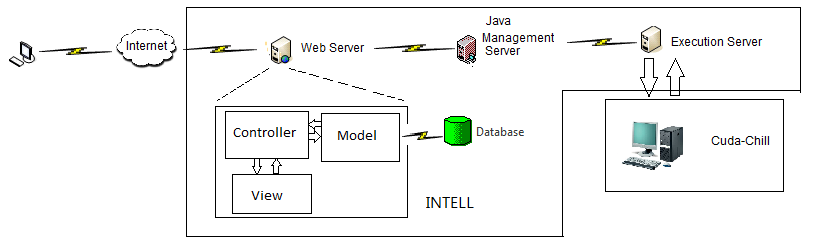
# SYSTEM OVERVIEW

INTELL shall utilize CUDA-CHILL framework to provide a web interface that would be used by researchers to optimize and test different benchmarks for different GPU architectures. It shall provide the researchers with test results of the optimization achieved under different configurations. This shall lead to better optimizations for code recipes used in software development.

# SYSTEM ARCHITECTURE

## Architectural Design

The basic requirement for INTELL is to provide a web interface that allows its users to utilize the functionality of the cuda-chill framework and benchmark different GPUs easily. In order to achieve these goals the design depends upon a multi tier client server architecture pattern.



INTELL is a multi-tier web application that is divided into 3 major modules.

* A web interface including a 3-tier web server
* A java web service available only to the web server
* A java job execution module which provides service to the Java Server.
  + 1. **A web interface including a 3-tier web server**

This shall provide a website interface through which the users shall interact with the system. It shall be a 3-tier web server in php utilizing the MVC( Model View Controller) architectural pattern. Thus it shall comprise of 3 subsystems or layers i.e Model, view and controller. Underneath the model shall be an SQL database for storing user information.

* + 1. **A java web service available only to the web server**

This shall be a backend java server that shall be responsible for handling the workload management for the GPUs available to the system. The web server would interact with this server through invoking a small module that shall send a job request to this server.The java server shall then designate a GPU to carry out the job and send a ‘job order’ to the java job execution module.

This server allows the system to separate the web and application portion of INTELL and thus allows the web server to operate independently.

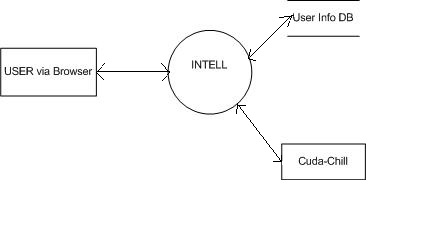
* + 1. **Java job execution module**

This shall be a small server module that shall provide the service of executing a job to the java server. It shall be deployed on each machine that has a GPU available for INTELL . Each of the instance of this module shall register itself with the java server on startup.This module is responsible for executing a job using cuda-chill and generating results in xml format.

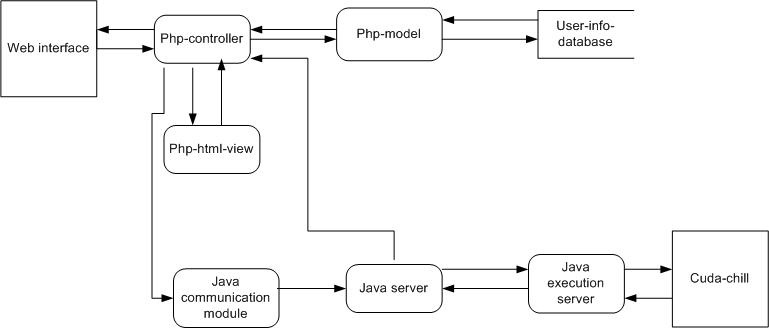
## Decomposition Description

# ****Data flow Diagrams****

1. **Top Level Data Flow Diagram**

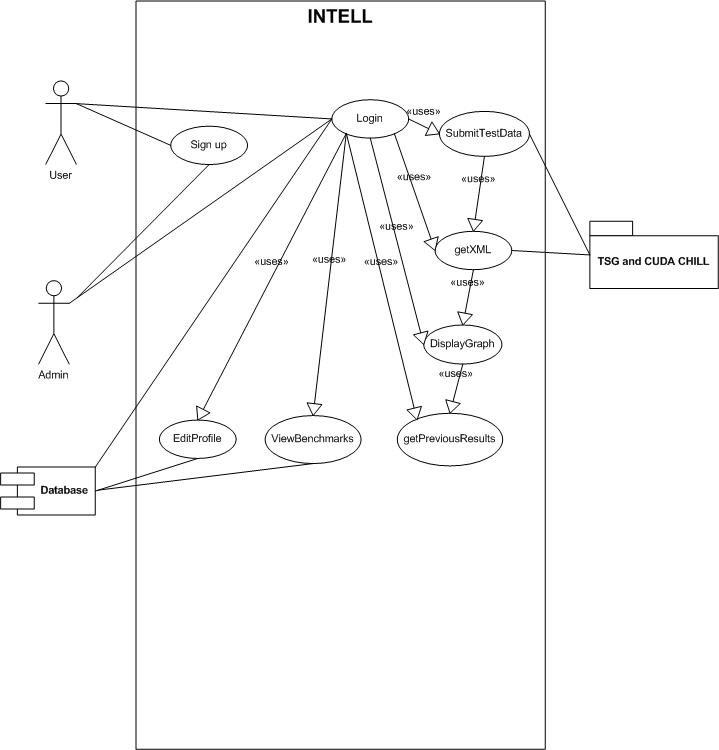


1. **Medium Level Data Flow Diagram**



## F:\Users\microsoft\Documents\use cases\Activity.jpgActivity Diagram

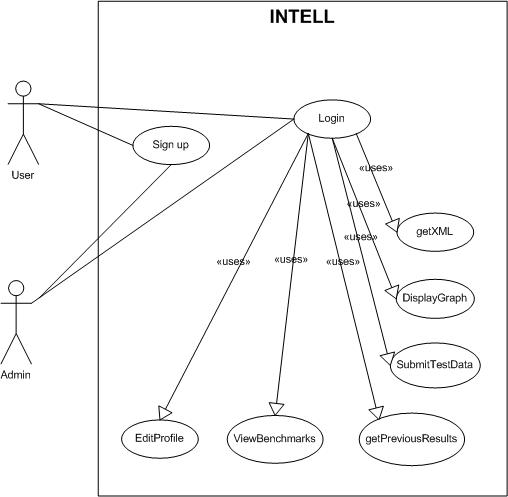
* + 1. **Use Case Diagrams**

****

**System wide use case Diagram**

1. **LOGIN**()

This Use Case describes the process by which users log into. It also sets up access permissions for various categories of users.



1. **Actors**

System User

Administrator

1. **Used Use Cases**

getXML()

DisplayGraph()

submitTestData()

getPreviousResults()

viewBenchamrksInfo()

editProfile()

1. **PseudoCode:**

The user goes to website.

User clicks login link which redirects him to login page.

The user enters his username and password and clicks submit

The system matches username and password combination with that from database.

If match is positive

    Allow user to use other page by storing a session variable with his identity.

Else

    Redirect user to login page.

1. **Pre Conditions**

The user must not be already logged in.

1. **Flow of Events**

The user request website by typing in URL of webpage.

User clicks login button which redirects him to login page.

The user enters his username and password and clicks submit.

The system gets username and password combination from database.

The system matches username and password combination with that from database.

If match is positive

Allow user to use other page by storing a session variable with his identity.

The system will set access permission.

The system will display main homepage to user.

Else

Redirect user to login page.

User will re-enter credentials for login scenario or leave web page by closing window.

1. **Post conditions.**

The user id session variable must not be destroyed before logout.

The user must be able to use services accessible to him. Including,

getXML()

DisplayGraph()

submitTestData()

getPreviousResults()

viewBenchamrksInfo()

editProfile()

1. **Alternate Scenarios**

**Bad username**

    The user is redirected to login page with message of invalid login details

**Bad password**

    The user is redirected to login page with message of invalid login details

**User does not have a username and password for our system**

     The user is asked to sign up first with our system.

**User selects a function for which they are not allowed access**

    The user is redirected to login page.

**User makes no selection of function**

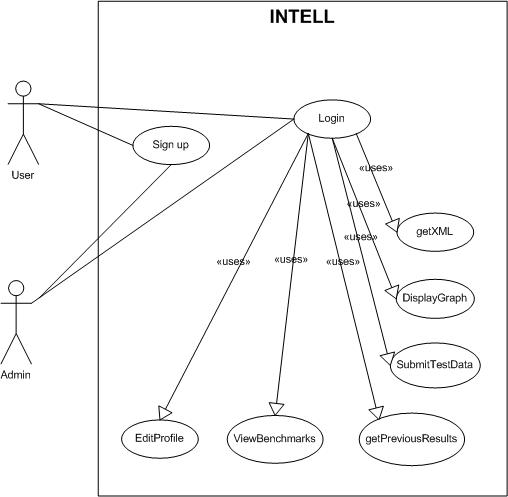
    The Home screen is display to user.

**User Logout**

The user logout is given as link on top right part of page and user session variable is deleted after logs out of system. The user is no more able to access system services.

1. **SIGN UP**

This Use Case describes the process by which users sign up with our system. It also sets up access permissions for various categories of users.



1. **Actors**

System User

Administrator

1. **Used Use Cases**
2. **PseudoCode:**

The user goes to website.

User clicks signup link which redirects him to signup page.

The user enters his details including name, mobile, email, institute and clicks submit button.

The system matches already existing emails for duplicates.

If match is positive

    User is redirected to signup page with error message displayed.

Else

    User is sent a verification email.

User clicks on link in email.

A code is sent to user mobile.

User is presented with a form to enter the code sent on their mobile phone.

If User enters correct code

Their profile is activated and they are redirected to login page

Else

They are asked to singup again since they were not able to verify their accounts. With proper message displayed.

1. **Pre Conditions:**

The user must not be logged in already.

1. **Flow of Events**

The user request website by typing in URL of webpage.

User clicks on sign up link which redirects him to sing up page.

Sign up page displays a message to user that it is 2 step authentication and they will be required to verify email and mobile number within same session.

The user enters his details including name, mobile, email, password, institute and clicks submit button.

User is redirected to signup page with error message displayed.

User is sent a verification email.

User clicks on link in email.

A code is sent to user mobile.

User is presented with a form to enter the code sent on their mobile phone.

Their profile is activated and they are redirected to login page

1. **Post conditions.**

The user is allowed to login using his username and password.

The user must be able to use services. Including,

getXML()

DisplayGraph()

submitTestData()

getPreviousResults()

viewBenchamrksInfo()

editProfile()

1. **Alternate Scenarios**

**Existing Email**

    The user is redirected to sign up page with message of email already exists.

**No Mobile Phone Code entered**

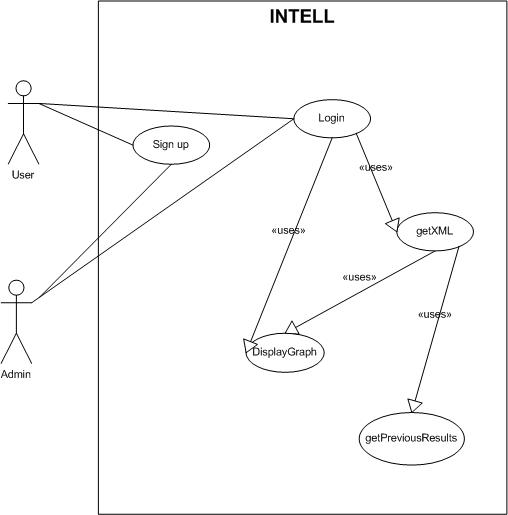
    The user is asked to sign up again and not allowed to login

**No email authentication link clicked**.

   The user is asked to sign up again and not allowed to login

1. **GetXML()**

In this use case, user requests for an XML version of results. The results are from tests performed on actual GPU based on inputs from user.



1. **Actors**

User

1. **Used Use Cases**

Login()

DisplayGraphs()

GetPreviousResults()

1. **Pseudo Code**

The user is already logged in

The user request for results of tests he requested on a GPU.

The database is looked up if the results are available or not.

If results are available then they are returned to user in XML structure.

Else the user is informed of the about their tests status that it is currently in wait queue for jobs.

1. **Pre Conditions:**

The user is already logged in.

The user has submitted test data.

1. **Flow of Events:**

SubmitTestData()

The user submits request for performing tests on GPU with benchmark and its required inputs to our system.

The user requests results of test by clicking on link in top menu.

The user is returned with result data in XML format which can be used for

DisplayGraph()

1. **Post Conditions:**

The XML follows standard format from w3c.

1. **Alternate Scenarios:**

**User has not submitted a test request.**

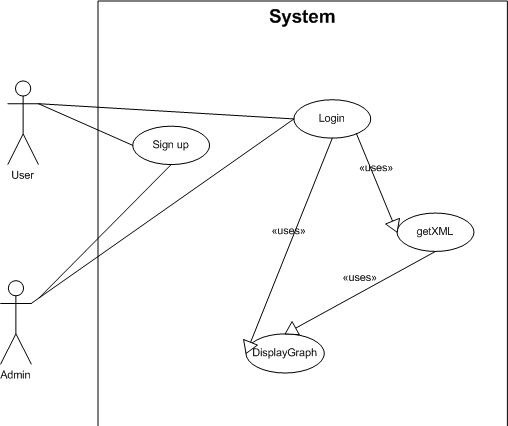
The user is displayed a message saying you must request a test before requesting results.

**Test is not completely executed**

The user is notified that test is in waiting queue and will be notified of results by email when its completed.

1. **DisplayGraph**

This use case implies displaying the data from getXML in graphical form.



**INTELL**

1. **Actors:**

User

1. **Used Use cases:**

Login()

GetXML()

1. **Pseudo Code:**

The user is already logged in.

The user has requested a test and the test is completely executed.

The user has received test results in form of XML.

The user request a graph of this test result.

The graphs are computed using JS and the XML data.

The user is able to view graphs and results later.

1. **Pre Conditions:**

User is Logged in already.

The user has requested test and received the test results.

1. **Flow of Events:**

The user requests to show graphs of data by clicking on link which is along with test results received.

The user XML data is used to create graphs with JS.

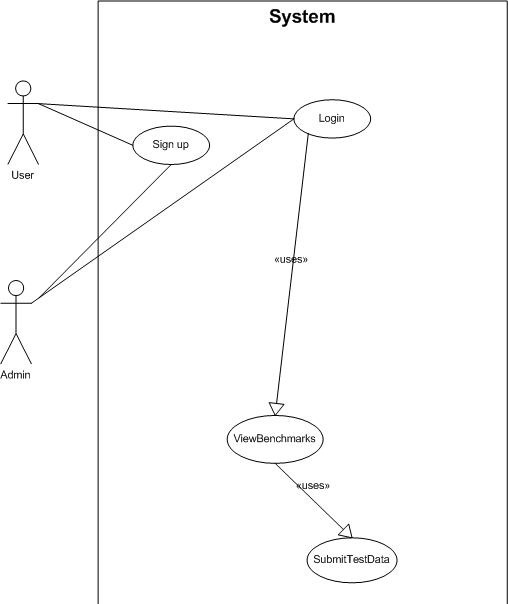
The graphs are displayed to user.

1. **Post Conditions**

User can view the graphs later.

1. **Submit Test Data**

This use case implies the user request for executing test on GPU by giving the benchmark requirements along with compulsory inputs for that benchmark.



**INTELL**

1. **Actors:**

User

1. **Use Cases used:**

Login()

1. **PseudoCode:**

The user is logged in.

The user clicks on link to perform test.

User selects a benchmark from drop down.

User selects input range for selected benchmark.

The user request is submitted and they are notified with results complete message as soon as these are completely executed and ready in XML form.

1. **Pre Conditions**

User is already logged in.

User has knowledge of Benchamrks and range of test data.

1. **Flow of Events.**

User is logged in to the system.

The user is presented with hyperlink on main menu to perform a test on GPU.

The clicks on link and is presented with a web page.

User selects a benchmark from drop down.

User selects input range for selected benchmark.

The user request is submitted and they are notified with results complete message as soon as these are completely executed and ready in XML form.

1. **Post Conditions:**

The users are able to view results submitted later.

The users are able to check on status of tests submitted.

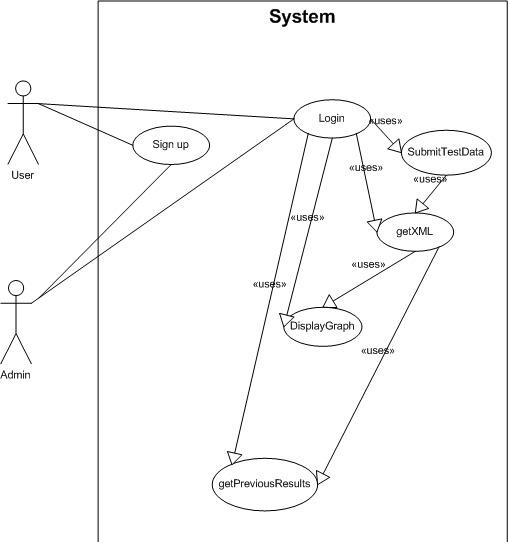
1. **Alternate Scenarios:**

**Corrupt Inputs:**

The user is requested to submit benchmark and input range again for performing test.

1. **View Previous Results**

This use case is for viewing results of previously submitted tests.



**INTELL**

1. **Actors:**

User

1. **Use Cases Used:**

getXML()

displayGraph()

SubmitTestData()

1. **Pseudo Code:**

The user is already logged in.

The user clicks on link for view past results for tests they have already submitted.

The DB is checked for any completed tests and their results.

If results are found

The user is returned with data in XML form.

The XML data is used to display graphs.

Else if results are pending

The user is notified that results are pending and they will be notified by email as results are available.

Else if no test submitted

If the user has not submitted any test then they are asked to request for tests.

1. **Pre Conditions:**

User has already logged in.

User has submitted test request with benchmark and respective inputs.

The test results are available to user.

1. **Flow of Events:**

The user is logged in.

The user request to view previous results by clcking on hyperlink on main web page.

The users test results are returned from system in XML form.

The XML returned is used to display results in graphs.

1. **Post Conditions.**

User is not allowed to delete any result.

1. **Alternate Scenarios**

**The user has not submitted any test request**

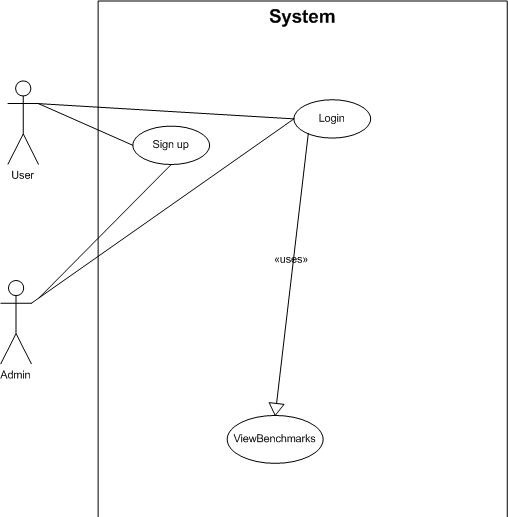
**The user is asked to submit a test request using SubmitTestData()**

**The results are pending**

The user is notified of current state that results are pending execution on actual hardware and they will be notified of results by email as they are available.

1. **View Benchmark Info**

This use case allows for viewing information about available benchmarks.



**INTELL**

1. **Actors:**

User

1. **Use Cases Used**

Login()

1. **Pseudo Code**

User is logged in and user requests for viewing benchmark details.

The user is presented with a dropdown of benchmark names.

User selects benchmark.

The user is presented with details about that benchmark

1. **Pre Conditions:**

User must be logged in.

1. **Flow of Events:**

The user logs in to the website using their username and password.

The user clicks on link to view benchmarks details from main menu.

The user is presented with dropdown to seletec a specific benchmark.

The user slected a benchmark.

The user is presented with details about the benchmark and input requirements for it.

1. **Post Conditions:**

The user is allowed to SubmitTestData()

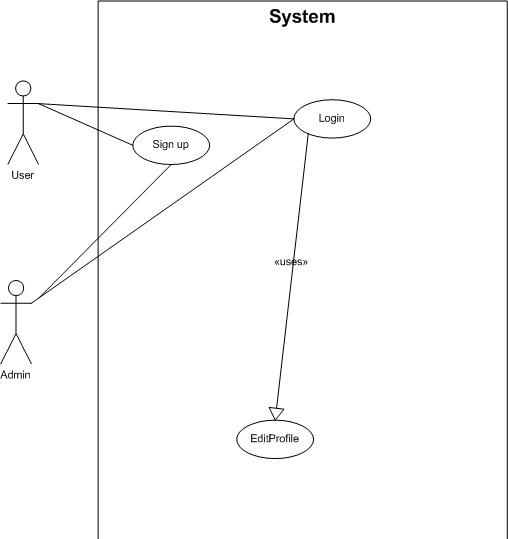
1. **Alternate Scenario**

**The user is not logged in.**

The user is not allowed to view benchmarks and details. They are redirected to login page.

1. **Edit Profile**

This use case allows for user to update their profiles.



**INTELL**

1. **Actors:**

Users

Admin

Use Cases Used:

Login

1. **Pseudo Code:**

The user selects to edit profile by clicking on link from main menu.

The user is presented with similar input form as of signup to edit their profile information.

If the user tries to update their email information, they are required to re verify their email by clicking on link in email and then entering the code send on mobile via input form.

If the user fails to enter code, the profile is not updated else they are notified of update on older email address.

1. **Pre Conditions:**

The user must be logged in already.

1. **Flow of Events:**

The user selects to edit profile by clicking on link from main menu.

The user is presented with similar input form as of signup to edit their profile information.

The user can opt in to update email or not.

On update of email, user is required to re verify their email by clicking on link in email and then entering the code send on mobile via input form.

The user profile is updated as code is entered.

The user is notified of change via older email as well.

Use case ends.

1. **Post Conditions:**

The user is able to access features accessible to him.

The user is able to login again.

1. **Alternate Scenarios:**

**The user fails to click on link in email**

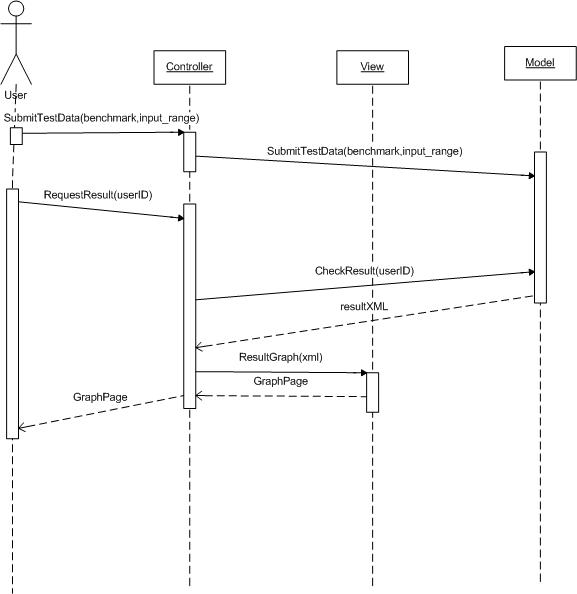
The user profile is not updated.

**The user fails to enter code on mobile.**

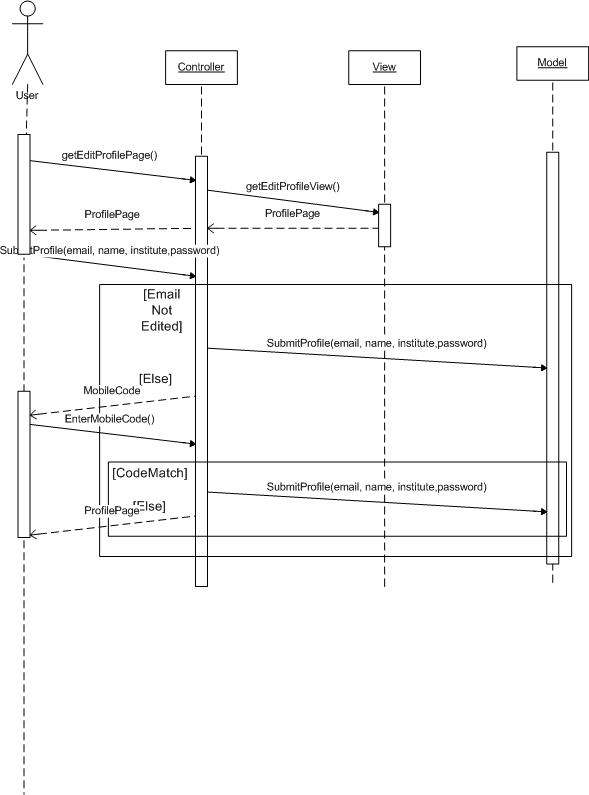
The user profile is not updated and notified of this particular activity via older email.

* + 1. **Sequence Diagram**

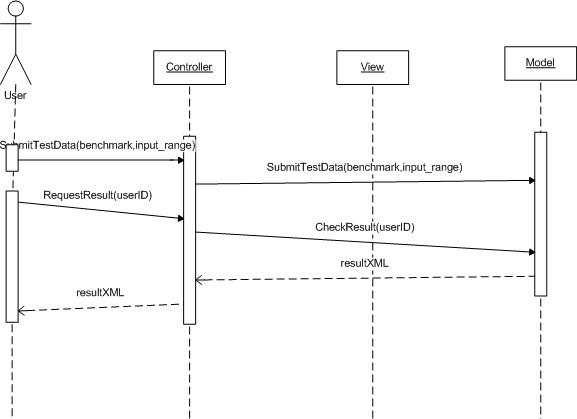
1. **Display Graph**

****

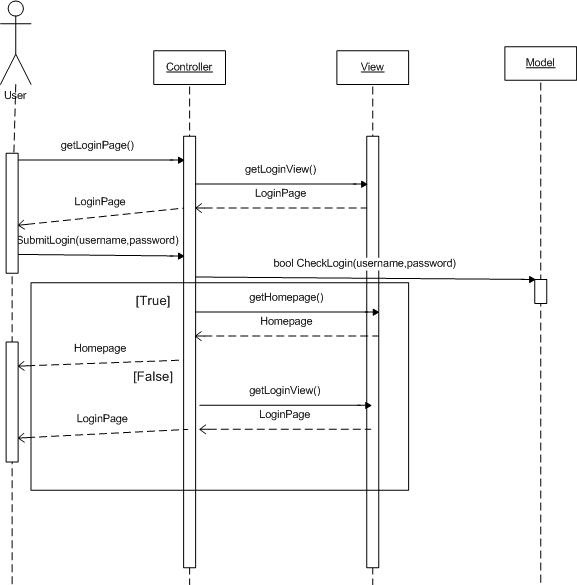
1. **Edit Profile**

****

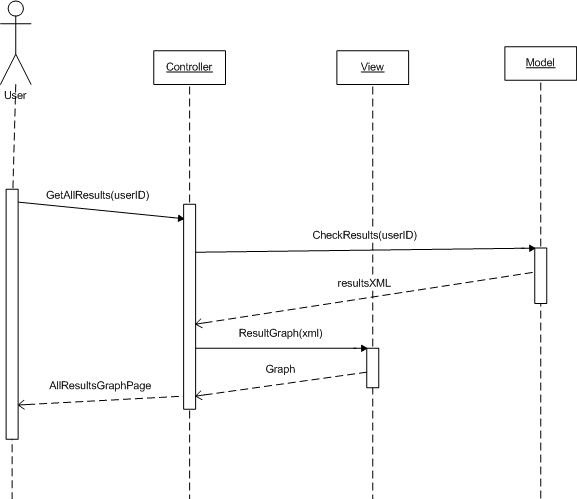
1. **getXML**

****

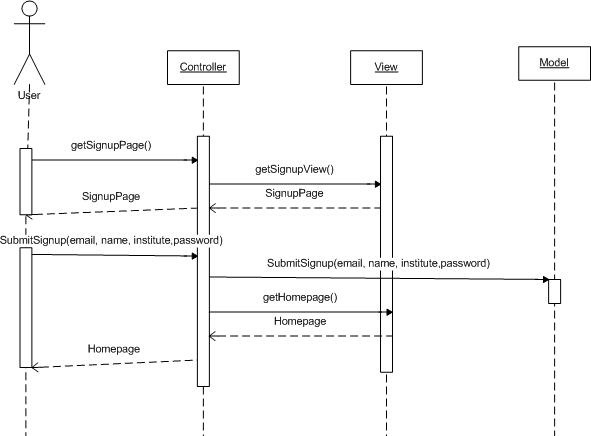
1. **Login**

****

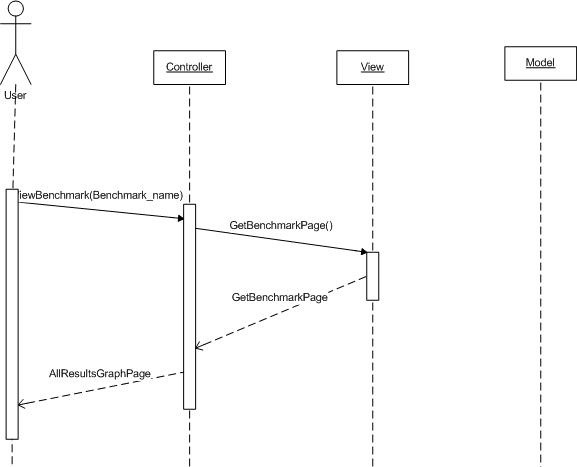
1. **Previous Results**

****

1. **Signup**

****

1. **View Benchmark**

****

## Design Rationale

The different design models that were considered are mentioned below.

The primary problem was the dual nature of the INTELL that is, it offers all the functionality of a basic website plus it also needs to actually execute code on hardware. In order to achieve this the problem was to bridge different technologies. The options available for this were

## Bridge php with c++

For this we had two options

1. To call an ‘executable request sending module’,we rejected this approach because allowing an executable to run on the server has too many security risks such as code injection into a process.

ii. To write a COM enabled custom dll and use it for IPC between a c++ module and php server. This approach had the factor of unnecessarily increasing the complexity of the system, also it still wouldn't solve the scalability problem because the c++ module would need to manage the execution of job request as well as communication with php server.

## Bridge php with java

For this the first option was to use a php/java bridge but this meant that the core kernel must be written in php but php is not meant for such complex tasks and the code becomes complicated and hard to maintain. Secondly it would mean that the php controller would be performing far too many tasks which means absolutely no scalability thus this approach was also rejected.

# DATA DESIGN

## Data Description

The important data structures that are required for this design are

1. Job queue in the controller

A jobQueue containing pending job requests.

Max queue length shall be 20.

Each JobRequest shall have following data.

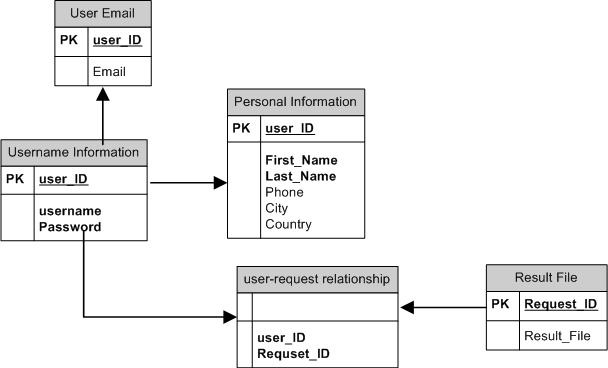
* reuest id
* username
* benchMarkName
* GPU model name
* benchMarkParameters as a string

1. A table containing info on status of each currently active instance of JavaExecutionModule , its available GPU model and its current status.

Columns for the table shall be

* Execution ModuleInstance ID
* GPU Model name
* Module Status as an enumeration

1. JavaExecutionModule shall maintain a string containing info on the GPU available to it.
2. The result generated shall be in the form of a well defined XML file.(see appendix for schema).
3. User information is stored in a sql database connected to the php-Model. Its ERD is provided below.



1. the information on the user’s chosen benchmark and parameters shall be transmitted using udp packets between different servers.

Each packet shall contain the following data as a payload

* + request ID
  + benchmark name
  + benchmark parameters
  + GPU model name

1. Controller maintains a pending job lookup table. a job request item is added to this table when its job request is forwarded to the java server.

Each job request item contains following data

* request id
* username
* benchMarkName
* GPU model name
* benchMarkParameters as a string

## Data Dictionary

* + 1. **Java Communication module**

1. **User Datagram Packet**

Payload Data

* + request ID
  + benchmark name
  + benchmark parameters
  + GPU model name

**Functions**

* SendUserRequest(int request ID, String benchmark name, String benchmark parameters ,String GPU model name)

## JavaServer Module

1. **DataStructures**

GPU Table

* Execution ModuleInstance ID
* GPU Model name
* Module Status as an enumeration

1. **Functions**

* RegisterExecutionGPU(Execution ModuleInstance ID, GPU Model name)
* SendJobRequest(requestID ,GPU\_model ,benchMarkName ,benchMarkParam)
* SendResultToController(xmlFileHandler,request ID)
  + 1. **JavaExecution Module**

1. **Datastuctures**

* String GPU Info

1. **Functions**
   * booleanExecuteScript(String benchmark,StringBenchmarkParam)
   * booleanExecuteScript(String benchmark,StringBenchmarkParam)
   * SendResultToJavaServer(String xmlpath)
     1. **PHP-Controller**
2. **DataStructures**

* JobQueue
* JobRequest shall have following data.
* request id
* username
* benchMarkName
* GPU model name
* benchMarkParameters as a string
* PendingJobRequest Table
* request id
* username
* benchMarkName
* GPU model name
* benchMarkParameters as a string

1. **Functions**

* DisplayGraph
* **EditProfile**
* **GetXML**
* **Login**
* SignUp
* Submit Test Data
* **View Previous Results**
* View Benchmark Info

# COMPONENT DESIGN

## PHP-Controller

## Module Description

This is the business logic component of the web server that utilizes the MVC architectural pattern. It is responsible for managing the user management constraints as well as forwarding the user’s job requests to the JavaServer Module.

## Module Specification

Since this module provides the basic services to the user it is composed of the following sub modules.

* 1. **Login**

**PseudoCode:**

The user goes to website.

User clicks login link which redirects him to login page.

The user enters his username and password and clicks submit

The system matches username and password combination with that from database.

If match is positive

Allow user to use other page by storing a session variable with his identity.

Else

    Redirect user to login page.

* 1. **SignUp**

**PseudoCode:**

The user goes to website.

User clicks signup link which redirects him to signup page.

The user enters his details including name, mobile, email, institute and clicks submit button.

The system matches already existing emails for duplicates.

If match is positive

    User is redirected to signup page with error message displayed.

Else

    User is sent a verification email.

User clicks on link in email.

A code is sent to user mobile.

User is presented with a form to enter the code sent on their mobile phone.

If User enters correct code

Their profile is activated and they are redirected to login page

Else

They are asked to singup again since they were not able to verify their accounts. With proper message displayed.

* 1. **GetXML**

**Pseudo Code**

The user is already logged in

The user request for results of tests he requested on a GPU.

The database is looked up if the results are available or not.

If results are available then they are returned to user in XML structure.

Else the user is informed of the about their tests status that it is currently in wait queue for jobs.

* 1. **DisplayGraph**

**Pseudo Code:**

The user is already logged in.

The user has requested a test and the test is completely executed.

The user has received test results in form of XML.

The user request a graph of this test result.

The graphs are computed using JS and the XML data.

The user is able to view graphs and results later.

* 1. **Submit Test Data**

**PseudoCode:**

The user is logged in.

The user clicks on link to perform test.

User selects a benchmark from drop down.

User selects input range for selected benchmark.

The user request is submitted and they are notified with results complete message as soon as these are completely executed and ready in XML form.

* 1. **View Previous Results**

**Pseudo Code:**

The user is already logged in.

The user clicks on link for view past results for tests they have already submitted.

The DB is checked for any completed tests and their results.

If results are found

The user is returned with data in XML form.

The XML data is used to display graphs.

Else if results are pending

The user is notified that results are pending and they will be notified by email as results are available.

Else if no test submitted

If the user has not submitted any test then they are asked to request for tests.

* 1. **View Benchmark Info**

**Pseudo Code**

User is logged in and user requests for viewing benchmark details.

The user is presented with a dropdown of benchmark names.

User selects benchmark.

The user is presented with details about that benchmark

* 1. **EditProfile**

**Pseudo Code:**

The user selects to edit profile by clicking on link from main menu.

The user is presented with similar input form as of signup to edit their profile information.

If the user tries to update their email information, they are required to re verify their email by clicking on link in email and then entering the code send on mobile via input form.

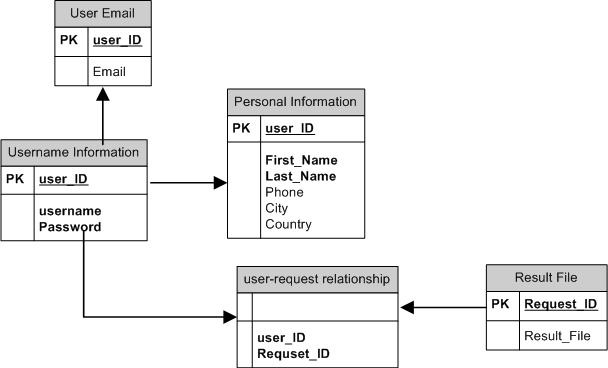
If the user fails to enter code, the profile is not updated else they are notified of update on older email address.

## PHP-Model

## Module Description

This is the data management component of the web server that utilizes the MVC architectural pattern .It is responsible for providing an interface for the Database to the controller and hiding the lower level detailed queries.

## Module Specification



## PHP-View

## Module Description

This is the view and display management component of the web server that utilizes the MVC architectural pattern. It comprises of a set of views that the controller can demand as required and then can be displayed to the user.

## Module Specification

Detailed specification of this module is covered under user interfaces in section 6.

## Java Communication module

## Module Description:

This module is to allow the php-controller module to send a job request to the java server module.

## Module Specification:

This module uses the SendUserRequest(int request ID, String benchmark name, String benchmark parameters ,String GPU model name) function to send the data received as parameters to java server module via a UDP payload. The required parameters are received from the controller module through command line upon invocation of this module, returns true if packet successfully sent.

begin

main requires reuest id, benchMarkName,GPU model name, benchMarkParameters as string arguments.

upon start up first parse the received arguments and store them in separate variables.

call the function

SendUserRequest(int request ID, String benchmark name, String benchmark parameters ,String GPU model name)

{

Now this function establishes a connection with java server

else throws error and returns false

create a UDP packet with the payload containing all the data recieved as parameters by this function.

send the packet

return true

}

exit process

end

## JavaServer Module

## Module Description:

The java server module separates the request management details from the web server.It allows multiple instances of javaExecution Module to register themselves. Then registered servers are given requests to execute and return the results.

## Module Specification:

This module has following sub modules.

## Register execution module

1. **Module Description**:

This is used to register an instance of java Execution module.

1. **Module Specification:**

When an instance of java execution module sends a registration request via a UDP packet to the java server this function is used to register i.e. add itsits information to the GPU table.

begin

this function is called by the java server upon recieng a regstration request by a java execution module.

RegisterExecutionGPU(Execution ModuleInstance ID, GPU Model name)

{

add a new row to the GPU table containg the data recieved as parametes by this function and then return true

else throw an error and return false

}

end

## SendJobRequest

1. **Module Description:**

Allows the java server to select and send a jobrequest to an appropriate instance of java execution module.

1. **Module Specification:**

Selects an instance of java execution module that matches the required GPU in the Job request with status entry set to available. It then sends a job request containing request id ,benchmarkname,benchmark parameters and Execution Module Instance ID of the recipient server module. It then updates the status entry of the selected instance to busy.

begin

SendJobRequest(requestID, GPU\_model ,benchMarkName,benchMarkParam)

{

searchgpu table for an available gpu that matches the model and return a reference to an object of the row containing the gpu entry

else throw an error and return false

If gpu found then establish a connection to the instance of java server module responsible for that GPU and send a job request via UDP payload

else throw an error

}

end

## SendResultToController

1. **Module Description**

Sends the results of a job as an xml to the controller module.

1. **Module Specification**

It first places the xml file containing the results in a predetermined location with the request id as the file name and then sends an http request to the controller containing the request ID and triggering an event in response to which the controller looksup the xml file and thus receives it.

begin

SendResultToController(xmlFileHandler,request ID)

{

Place the XML file in a predetermind default directory to whci the controller has access with the request\_ID.xml as file name

then send a lookup request to controller via HTTP packet

}

end

## JavaExecution Module

## Module Description

It is responsible for executing the job requests via cuda-chill through invoking a python script.Also it must acquire the details of the on board GPU upon start up and then send a registration request to the java server module.

## Module Specification:

1. **booleanExecuteScript(String benchmark,StringBenchmarkParam)**

{

Execute a python scrept using exec command of Java Process object with

a string array containing the parameters provided to this function

}

## Generate xml

{

generate an xml iteratively from the raw data file generated by the python script

}

## SendResultToJavaServer(String xmlpath)

{

create a connection to the java server and then send the xml file along with instance id via UDP payload.

}

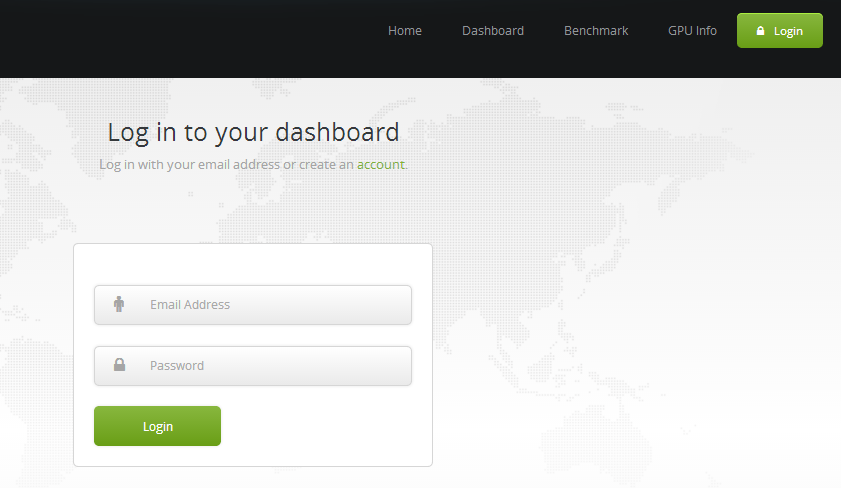
# HUMAN INTERFACE DESIGN

## Overview of User Interface

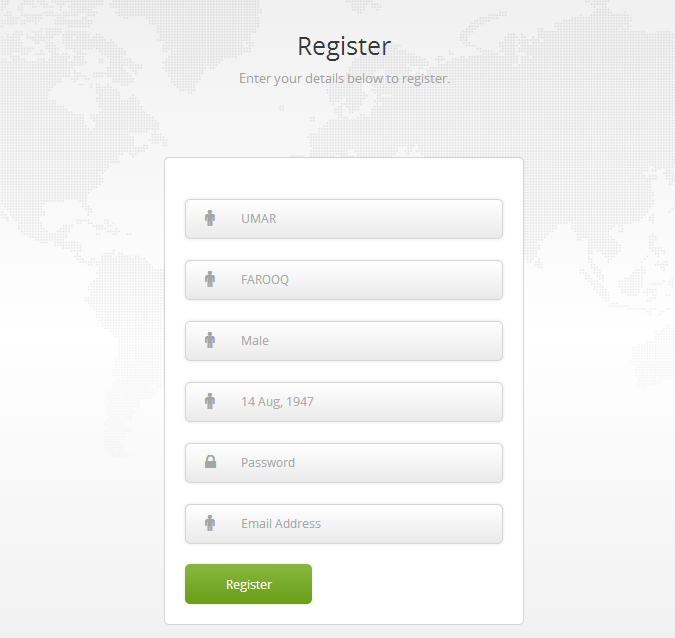
Describe the functionality of the system from the user’s perspective. Explain how the user will be able to use your system to complete all the expected features and the feedback information that will be displayed for the user.

## Screen Images

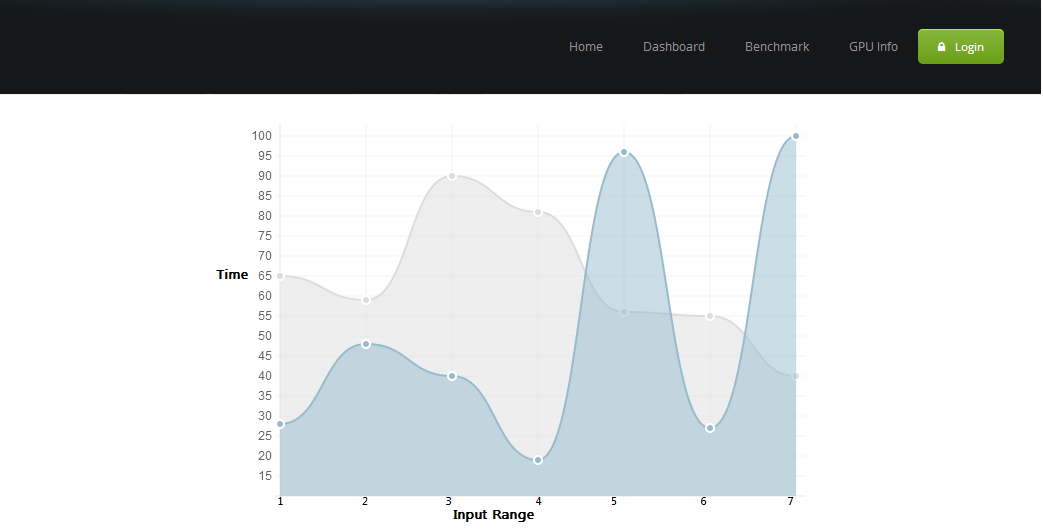
1. Login



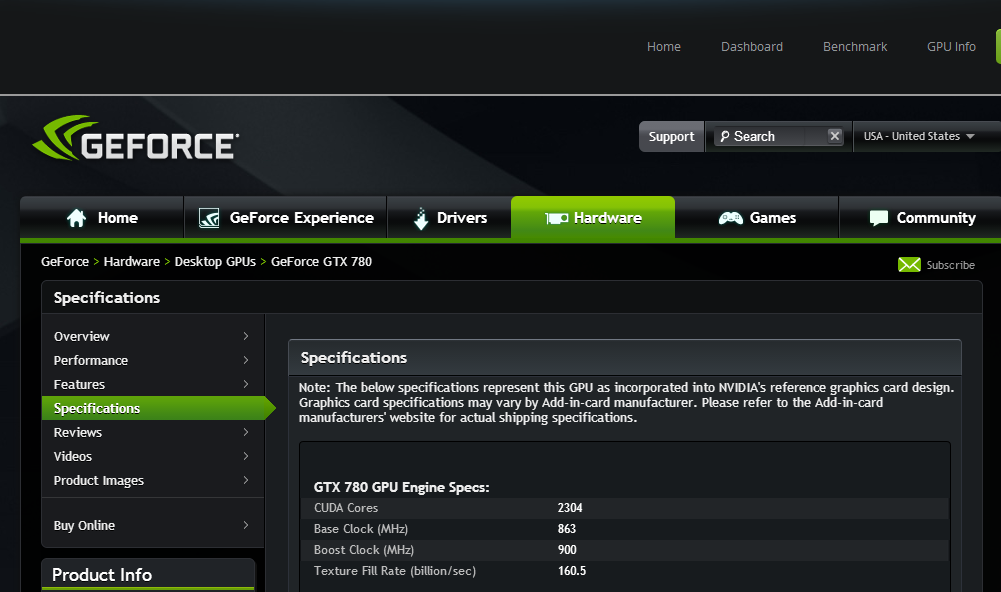
1. Signup



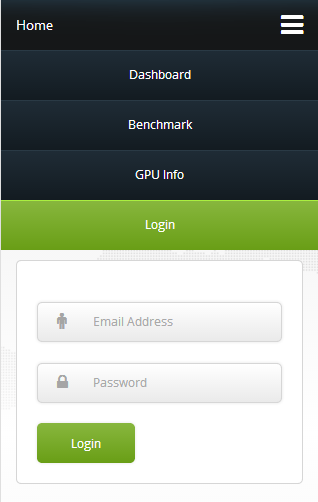
1. Display Graph



1. Hardware Specification



1. Mobile Version



## Screen Objects and Actions

The interface is designed to be simple and easy to use so basic elements on each page are well defined and prominent. The color scheme used has high contrast so that the visual objects are easily recognizable.

The user actions are already covered under use cases, the user is allowed to sign up ,login browse benchmarks,browse GPU info , request benchmark tests and view results.

# REQUIREMENTS MATRIX

Provide a cross­reference that traces components and data structures to the requirements in your SRS document.

Use a tabular format to show which system components satisfy each of the functional requirements from the SRS. Refer to the functional requirements by the numbers/codes that you gave them in the SRS.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Requirements\Modules | PHP Model | PHP Controller | PHP View | Java Comm | Java Server | Execution Server |
| FR1 |  |  |  |  |  |  |
| FR2 |  |  |  |  |  |  |
| FR3 |  |  |  |  |  |  |
| FR4 |  |  |  |  |  |  |
| FR5 |  |  |  |  |  |  |
| FR6 |  |  |  |  |  |  |
| FR7 |  |  |  |  |  |  |
| FR8 |  |  |  |  |  |  |
| FR9 |  |  |  |  |  |  |
| FR10 |  |  |  |  |  |  |
| FR11 |  |  |  |  |  |  |
| FR12 |  |  |  |  |  |  |
| FR13 |  |  |  |  |  |  |
| FR14 |  |  |  |  |  |  |
| FR15 |  |  |  |  |  |  |
| FR16 |  |  |  |  |  |  |
| FR17 |  |  |  |  |  |  |

# APPENDICES

**Appendix A**

**XML Schema**

<?xmlversion="1.0"?>

<xs:schemaxmlns:xs="http://www.w3.org/2001/XMLSchema"targetNamespace="http://www.intell.local"xmlns="http://www.intell.local"elementFormDefault="qualified">

<xs:elementname="Result">

<xs:complexType>

<xs:sequence>

<xs:elementname="BenchMark">

<xs:simpleType>

<xs:restrictionbase="xs:string">

<xs:enumerationvalue="Matrix-Matrix-Multiplication"/>

<xs:enumerationvalue="Matrix-Vector-Multiplication"/>

</xs:restriction>

</xs:simpleType>

</xs:element>

<xs:elementname="ProblemParameters">

<xs:complexType>

<xs:sequence>

<xs:elementname="start\_size"type="xs:decimal"/>

<xs:elementname="end\_size"type="xs:decimal"/>

<xs:elementname="step\_size"type="xs:decimal"/>

</xs:sequence>

</xs:complexType>

</xs:element>

<xs:elementname="GPU\_Info"type="xs:string"/>

<xs:elementname="CPU\_info"type="xs:string"/>

<xs:elementname="RAW\_Data">

<xs:complexType>

<xs:sequence>

<xs:elementname="row"maxOccurs="unbounded">

<xs:complexType>

<xs:sequence>

<xs:elementname="matrix\_size"type="xs:decimal"/>

<xs:elementname="GPU\_RunTime"type="xs:decimal"/>

<xs:elementname="CPU\_RunTime"type="xs:decimal"/>

</xs:sequence>

</xs:complexType>

</xs:element>

</xs:sequence>

</xs:complexType>

</xs:element>

</xs:sequence>

</xs:complexType>

</xs:element>

</xs:schema>

**Appendix B**

**Sample XML**

 <?xmlversion="1.0"encoding="utf-8" ?>

<Resultxmlns="http://www.intell.local"xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"xsi:schemaLocation="http://www.intell.local reslt\_schema.xsd">

<BenchMark>Matrix-Matrix-Multiplication</BenchMark>

<ProblemParameters>

<start\_size>1024</start\_size>

<end\_size>8192</end\_size>

<step\_size>512</step\_size>

</ProblemParameters>

<GPU\_Info>Nvidia-GTX-770</GPU\_Info>

<CPU\_info>Intel(R)-Core(TM)i5-4430-3.00GHz</CPU\_info>

<RAW\_Data>

<row>

<matrix\_size>1024</matrix\_size>

<GPU\_RunTime>1.05612378</GPU\_RunTime>

<CPU\_RunTime>3.065412378</CPU\_RunTime>

</row>

<row>

<matrix\_size>2024</matrix\_size>

<GPU\_RunTime>2.05612378</GPU\_RunTime>

<CPU\_RunTime>3.565412378</CPU\_RunTime>

</row>

<row>

<matrix\_size>3024</matrix\_size>

<GPU\_RunTime>1.15612378</GPU\_RunTime>

<CPU\_RunTime>4.065412378</CPU\_RunTime>

</row>

<row>

<matrix\_size>4024</matrix\_size>

<GPU\_RunTime>1.25612378</GPU\_RunTime>

<CPU\_RunTime>4.565412378</CPU\_RunTime>

</row>

</RAW\_Data>

</Result>