Horizon Analytics

Technological Feasibility

10/26/2017

Sponsor: Harlan Mitchell, Sys. Tech. Manager

Honeywell Aerotech

Mentor: Austin Sanders

Project Members:
Emilio Sifuentes
Josh Baker
Rex Rogers
Summer Stapleton

Table of Contents

1. Introduction

The project under discussion within this document is the Next Generation BTU (Bench Test Utilities) Proof of Concept, and our client is Harlan Mitchell, a Systems Tech Manager for Honeywell Aerotech. This project consists of a web-based application system that involves the ability to standardize script generation, interact with a database containing the test scripts for the testing of Engine Control Units via the BTU, and store and analyze the results from these tests. The issue with the current system, despite its current working state, is that it is outdated in its current format both in practical usage and in the current state of its hardware and software. In addition, testers and engineers must access this system from designated machines that are physically located in the same room as the BTU itself. This document is for discussing the options available for developing an updated, platform-independent application that the users may access from their own computers. It is essential to create BTU Concept that is able to fulfill the requirements laid out by our client. This document is organized into sections involving the challenges faced, approaches to these challenges, integration of the chosen paths, and a brief conclusion that will summarize the plan of action.

Each of the sections in this document describes a technological aspect within the system being developing. In the “Technological Challenges” section, the hurdles faced when it comes to general development and where focus needs to lie in order to make sure the whole project actually works are specified. The “Technological Analysis” section focuses on the choices to be made, the approaches being taken with this project, and the feasibility of those approaches when it comes to networking, database systems, as well as front and backend frameworks and languages. “Integration” will be the section that brings our solutions together since they will all be interacting on some level. Finally, our “Conclusion” section will summarize our research.

This is a Requirements Engineering Capstone Project for The College of Engineering at Northern Arizona University for the 2017-2018 academic year. The Capstone Team name is Horizon Analytics and consists of: Emilio Sifuentes, Josh Baker, Rex Rogers, and Summer Stapleton.

2. Technological Challenges

The act of refactoring a legacy system presents several challenges. Starting from details on what the current systems layout is and ideas on how to implement them in a more accessible format, a list of the technical challenges has been produced that need to be overcome in order for this project to appropriately fulfill our client's needs. Below is a list of technical challenges that are related to those requirements. The system must:

- Be accessible to engineers regardless of operating system
- Be able to store relevant engine simulator tests
- Allow engineers access to engine simulator tests
- Have a system that generates test reports
- Implement networking to transfer files
- Have the ability to standardize scripts
- Have a database for the stored tests

In order to address these challenges, a web-based application that may be accessed regardless of operating system and from a user's own machine will be implemented. Since this application needs to be dynamically responsive to user's input, a database needs to be implemented, as well as establish communication between the two. The application also needs to create automated testing scripts and collect, analyze and store results that have been returned to it back into the database. Finally, this application needs to communicate with the BTU system via the simulator in order for these test to be run and results returned.

This process is divided into three main objectives:

2.1 The Network

The proposed solution will involve communicating between various systems, a well-constructed network will be the backbone of our project. The goal here is to allow the server to deliver python script files to the simulator, and allow the simulator to respond by sending a test result HTML file back to the server. The goal will be quick and efficient communication between the server and simulator.

2.2 The Database

In order to allow a user to store, retrieve and perform calculations on information, the application must be able to do the same. This is why our solution will include a database. This database must allow for quick and accurate insertion and requests. It will store test scripts, test results, parameters for tests, and possibly user information.

2.3 The Application

A web-based applications can be broken into the front-end/client-side and the back-end/server-side. In order to create a successful application, these two pillars will need to be fully implemented both individually and together as a unit.

The client-side implementation will be what users will see and interact with, and so it is important to create a simple, powerful and responsive design that will be a delight to use.

The server-side implementation will need to be involved in all the back-end communications and operations in order to deliver something of use to users at the front-end. It can be seen as the middle-man of sorts since it will need to communicate will all parts of our system. It needs to read user inputs through the client-side and deliver accurate and meaningful responses, it needs to pull information from and store information to the database, and it needs to send the simulator scripts for testing as well as read the results back in from the BTU.

Communication with Front End	Communication with the Database	Communication with the BTU/Simulator	Internal Procedures
● Receive user inputs ● Send calculated/ meaningful outputs	● Retrieve necessary information ● Store scripts ● Store test results	● Send test scripts ● Receive test results	● Automate test scripting ● Run calculations on test results ● Create meaningful reports

Table 1: Responsibilities of the Application Back-End

3. Technological Analysis

In this section, some of the options and tools that are available to us for the development of our solution is explored. As discussed above, there are three key points that need to be addressed in our product: the network to tie the systems together, the database to store pertinent information, and the web-based application that users may use to interact with the Bench Test Utility.

3.1 Network Analysis

As our application will involve the integration of many systems and machines, the implementation of network protocols will be needed to allow for proper communication to allow test scripts and test results to be sent between the server and the simulator.

3.1.1 Transport Layer Protocol

The first research discussion relating to networking is what transport layer protocol to use, since it is the lowest networking layer that has a choice to be made when deciding how to use it. The real two options here are User Datagram Protocol (UDP) and Transmission Control Protocol (TCP).

UDP uses connectionless communication to send datagrams over a network. Each datagram is individually addressed and sent to the recipient. Due to its connectionless nature, there is no checking to guarantee that datagrams have arrived, nor any way to guarantee an order of arriving datagrams. UDP's main strength is its lower overhead; not having to resend lost information and not requiring a connection to be established to send data while also having a smaller packet header size speed up the process. This makes UDP a good fit for time-sensitive or delay-sensitive applications for things such as video streaming and voice-over-IP.

TCP uses connection based communication to send segments over a network. Each segment is numbered to place it in order with the other segments. The client then requests the server to resend any missing segments. This is TCPs main strength, reliable delivery of data; information reaching the client is guaranteed and can be put back into the same order it was originally in.

The primary purpose of the networking in this project is to transfer files containing scripts to be run from the server to the simulator. When transferring files it is important for the entire thing to be delivered, otherwise what is received isn't of much use, so TCP is the clearly the correct decision here.

Protocol	Packet Header Size	Connection Oriented	Reliable Delivery	Ordered Delivery	Data Checksum	Checksum Size
TCP	20–60 bytes	Yes	Yes	Yes	Yes	16 bits
UDP	8 bytes	No	No	No	Optional	16 bits

Table 2: Transport Layer Protocol Comparison

3.1.2 Application Layer Protocol

After selecting TCP as this project's network layer protocol, an application layer protocol that utilizes TCP to further simplify the process of sending our test script files can be chosen. Application layer protocols specialize in transferring files specifically, and there are quite a few options compared to at the network layer. These protocols help by establishing communication protocols, including rules and syntax for sending files, as well as the methods specific to communicating over established connections. Three of the most widely used applications layer protocols are considered and in order to determine the best fit for this project.

- **FTP**

  File Transfer Protocol is usually the first protocol of this type that comes to mind when file transfers are needed. It is a fairly old protocol, not having been updated for decades, but was originally designed with the specific purpose of sending files over a network, as the name suggests. When a transfer is started using FTP a random port is selected to send the files over; this can lead to NAT gateways and firewalls blocking the transfer if they do not trust the port that was selected. Due to the way ports are randomly selected it is possible for FTP to create port-stealing issues, where the selected port was already in use resulting in another legitimate connection being usurped. Requesting a file via FTP involves a series of commands from the client and responses from the server, requiring several round trip messages in order for a single transfer to complete. All major web browsers support FTP, which means there should be no compatibility issues on the user end when using this protocol. Another thing to note is that FTP transfers are not secure, none of the sent data is encrypted. There is FTPS, which incorporates SSL into the protocol to provide encryption and establish secure connections, but has the issue of not being supported across all browsers.

- **HTTP**

  Hypertext Transfer Protocol's original purpose is to be the data communication foundation of the world wide web, helping transfer the HTML and CSS files that are required to display web pages. Its purpose led to it fixing some latency issues that FTP had with sending multiple small files, which is typical when requesting a website. HTTP accomplishes this via persistent connections, allowing multiple transfers to happen over the same connection, cutting out additional 3-way handshakes. HTTP pipelining further speeds up the process by allowing multiple requests to be queued rather than requiring one request to finish before the next can be made. Yet another way HTTP is faster compared to FTP is that they take a single request and response to initiate and complete a transfer, further minimizing overhead. Compression is natively supported through HTTP, allowing server and clients to select from a range of compression options to further minimize the amount of data that needs to be transferred which can even further reduce the time to transfer. HTTP also always uses well known port numbers, 80 and 8080, which firewalls and NAT gateways will allow through without issue. Like FTP, HTTP is not inherently secure but has a secure version that utilizes SSL encryption called HTTPS, however, this has little to no browser support.

- **WebDAV**

  Web Distributed Authoring and Versioning is a protocol that is built on top of HTTP, expanding upon the options that HTTP has itself. WebDAV thus has mostly the same advantages and disadvantages that HTTP has, including having a secure version, WebDAVS, that isn't well supported. What WebDAV brings to the table over HTTP is that it adds filesystem-like support, it adds a framework through which users can create, edit, delete, and move files on a server. It also maintains property information about particular files, such as authorship, modification dates, collections, and overwrite protection. Many operating systems have built in client-side support of WebDAV. The additional features of WebDAV lend it to projects where joint authorship of documents is required.

Protocol	**Persistent Connection**	**Pipelining**	**Requests per Transfer**	**Compression Support**	**Consistent, Well-Known Ports**	**Filesystem Support**
FTP	No	No	Multiple	No	No	No
HTTP	Yes	Yes	1	Yes	Yes	No
WebDAV	Yes	Yes	1	Yes	Yes	Yes

Table 3: Application Layer Protocol Comparisons

3.1.3 Conclusions: Network

For the purposes of this project test scripts and test results need to be sent over a network, sometimes several at a time. What this means is that while WebDAV is a more robust version of HTTP, the extra features it provides are unnecessary. Filesystem-like support is not going to be helpful in this project and will simply add more complexity to the system than is required. When comparing HTTP to FTP it becomes clear that FTP is a dated protocol. FTP has the potential to cause issues with any firewalls that may be present on our clients networks, while HTTP avoids this issue entirely. HTTP also handles sending multiple files much more elegantly via persistent connections and pipelining, which is a feature our solution will need to implement. Finally, compression support and using a single request per file transfer further speeds up HTTP. The superior choice in regards to this project is thus **HTTP** due to its ability to transfer multiple files efficiently.

3.1.4 Solution Feasibility and Testing: Network

In order for this project to succeed, multiple python script files need to be able to be sent over the internet from the server to the simulator, and the simulator needs to be able to send the test result HTML file back to the server. To demonstrate that the HTTP is a feasible solution to this problem, a simple demo that will upload 10 queued sample python scripts over a HTTP connection from a server to a client will be developed. The client will then respond by sending a sample HTML file back to the server over a HTTP connection. This accomplishment will demonstrate HTTP as a good choice for the networking portion of the project.

3.2 Database Analysis

Considering Databases for this specific project requires some analysis of the requirements for application. From the technological challenges applicable:

- The engine tests seem that they will be similar in data configuration since they will all be tested for flying conditions in which test output should be around the same and input will probably also be the same.
- The database will need to be accessible by the engineers
- The tests within the databases will need to be easily manipulated either in an out from the database or within the database itself

There are several options considering that the requirements do not currently lend themselves to specific needs. The most notable three database options for this project were Oracle, Microsoft Access, and MySQL. From the current choices:

- **Oracle**

  Oracle is a bit expensive to implement as a database in terms of time and work. It requires a longer setup period in order to get started. The package itself is costly financially and considering what Honeywell would have to set it up for on a large level, it might be a difficult change in updating what they already have. However, since the engineers are performing many tests on each engine, the ability to handle multiple users on the same database depending on how many engineers will actually be accessing it at once would be useful. Also, since the database itself is a little more difficult to implement at the starting stage and to develop specifically for it, allows Oracle to be a bit more sturdy. Another point in Oracle's favor is the large amount of data it can handle, but so far the current iterations of data are test specific. So, large data handling would be useful, but it might not be exactly what is needed for this specific database. On top of everything else, there is a small amount of experience present within the project team in Oracle as a database.

- **Microsoft Access**

  Currently, the Bench Test Utilities that Honeywell is using is Microsoft Access 2003. Which means it is possible to update to a more current version of this software, should that be the final choice. Firstly, the package would be cheap compared to most other database options. Another good note, updating from Access 2003 should be relatively simple along with the engineers having experience in Access as a database already. Microsoft Access is normally better for single person use. All the data is accessible by a single file and it cannot handle large amounts of data. Since, they are already using Access the data amount might not be extremely pertinent, but it would still benefit them if the data amount they do have might benefit more from a database that can handle larger amounts of data. Considering that the reports and the tests are different sets of data, the single file aspect might be a bit more inconvenient than it needs to be.

- **MySQL**
  MySQL itself is free software, but certain packages for large commercial use might need to be purchased. This is an advantage cover the alternatives, but may not be the best selling point on this database since the client would most likely end up using a paid version of MySQL. In MySQL, you have to spend a bit of time to do things that other databases just do. Unlike the other alternatives, the project team has morel experience in MySQL then the alternatives. MySQL itself has plenty of user interfaces available for use. If necessary, it could also work in tandem with Oracle should that be a choice made in the future regarding where and how the tests are stored along with their data. In regards to other databases available, MySQL is a little less secure.

Database	Pros	Cons
Oracle	● Handles Multiple users well ● Development is more involved ● Robust ● Handles large amounts of data	● Expensive packages ● Longer Setup
MySQL	● Cheap option ● User Interfaces widely available ● Works in tandem with other databases in cases that may be applicable ● Fairly well rounded ● Familiarity with development	● Less Secure than alternatives ● Some functions take a little more work
Microsoft Access	● Cheap package ● Already using older version ● Engineers have experience with it to a degree	● Usually meant for single person use ● Single file storage ● Does not handle as large amounts of data as other options

Table 4: Database Comparison

3.2.1 Conclusions: Database

From the current choices, the decision is for **MySQL database**. Having a well rounded database considering the amount of data that will be handled and the ability to work in tandem with another database should the need arise in convenience for the engineers and the client. Familiarity with MySQL and the development language will allow better handling of the building and development of the database at a better pace as well. User interfaces being readily available is also quite the plus to this project.

As the database is developed, the datatypes that are handled should be deeply considered. Using the data given by the client will allow the testing of actual data being handled by the database within the programs being made in tandem. Most likely, testing engineer interaction first to make sure it is functional in that regard then focusing on how accessible the data within it is in terms of manipulation and consistency. The database will need to have functions that allow it to send its data or have its data viewed by the tester itself in order to generate the required reports. Each of these testing phases will be necessary to making sure this database is what it needs to be.

3.2.2 Solution Feasibility and Testing: Database

The database will implement a basic database using **MySQL** that is meant to handle one step of the data at a time before implementing the entire database at once. This will involve setting up sets of basic numbers that must be manipulated using the application side once it is developed enough to be a part of manipulating a database.

Once the connection from the application to the database is confirmed, testing can move on to more complex sets of data that resemble the tests that Honeywell currently uses on the simulators when performing tests. After those sets of data can be manipulated from the application, implementation of further functionalities required of the database can further develop uses that the database fulfills.

Using similar methods of testing, evaluation of the database choice with the retrieval of important values from the database tables. This will allow development of the database choice into the intended functionality of storing tests, manipulating those tests, retrieving data for those tests, and storing performance reports.

3.3 Application Analysis

The ultimate goal of this project is to have a platform-independent web-based application that Honeywell's testers can use in their Bench Test Utilities system. This application will need to consist of both client-side as well as server-side implementation.

3.3.1 Client-Side Frameworks

The client-side of the web based portion of this product will be the primary interface for the users. This portion of the product will need to interface with a server based program, as well as provide an easy to use, reliable interface. This interface will need to be well formatted and operate on many different systems and browsers. The languages being used to create this web based application will be HTML, CSS, and JavaScript. These three languages will need to be integrated in a way that will be effective and meet the requirements of the project. In order to accomplish this, a JavaScript framework will be utilized that will integrate the HTML, CSS, and JavaScript in a single development environment and allow for the creation of a reliable, robust web based application. AngularJS and EmberJS, are both viable options for creating the client side of the web based application.

- **AngularJS**

  AngularJS is an open source JavaScript framework that was created by Google. Because this framework is open source, there exists the option to replace pieces of the framework with our own code if needed. Use of AngularJS has grown greatly over the past few years and continues to grow. There is a large online community for AngularJS which will make it easy when it comes to finding more information about a particular aspect of the web application. AngularJS utilizes the MVC or Model, View, Controller design. Several documents were reviewed detailing client server interaction involving AngularJS. A drawback to AngularJS is that it is an older framework but this is made up for by extensive documentation. AngularJS has been proven to be reliable, robust and capable of meeting our needs for this project.

- **EmberJS**

  EmberJS is an open source JavaScript framework developed by the Ember Core Team as well as individual contributors. Several large companies such as Netflix, LinkedIn and even Microsoft use this framework for their websites. EmberJS uses the MVVM or Model, View, Viewmodel design. Being an opinionated framework, meaning it is built around a specific design pattern, it is easy for future developers to update or modify the application if they have knowledge of the framework. This framework has a large number of tools that can save an experienced EmberJS developer time, but can add to learning time for new developers. EmberJS appears to be a more powerful alternative to AngularJS but this power comes at a price, EmberJS has a steeper learning curve.

Framework	Learning curve	Design	Popularity	Testing	License	Opinionated
AngularJS	Medium	MVW	Approximately 1300 contributions on GitHub	Yes, unit tests as well as mock services	MIT	Flexible
EmberJS	High	MVVM	Approximately 500 contributions on GitHub	Yes, Mock services	BSD-3-clause	Strict

Table 5: Client-Side Framework Comparisons

3.3.2 Server-Side

A server-side application is essential in the deployment of dynamic web pages, such as will be displayed on our front-end/UI. As server-side application development can be tedious by hand, and to help significantly reduce development time, the decision has been made to make use of one of the many open-source web application development frameworks available. A framework is the structure behind client-server communication as well as database integration, and helps to specify output based on user input.

In addition to client and database communications, our server-side application must be able to produce scripts to send to the simulator, as well as store and analyze test results returned from it. While the communication between application and simulator has been addressed in the “Networks” section, the issue of creating these scripts must still be addressed.

3.3.2.1 Languages

While there are a few options when it comes to scripting languages that run server-side, there are two that definitely stand above the rest in implementation and popularity.

- **PHP**

  PHP: Hypertext Preprocessor is an open-source, widely used and powerful scripting language that is commonly used in web development applications. PHP was established in 1995 with the shift toward a more dynamic web experience, and so is well established in the web development community, with a large and active user base.

  Though powerful in application, PHP has been known to possess a steeper learning curve and does not make some of the more advanced capabilities easy for a beginner to grasp.

- **Python**

  A widely used high-level, general-purpose programming language geared toward more intuitively implemented and easy to read code. Python has been growing in popularity within the web development community as Python scripts have been found to work exceptionally well in this context.

  Our client already implements their testing scripts in Python, and so this option may be easier for future software maintenance.

Language	Learning Curve	Already In Use on Client System	Established Community	Includes Popular Frameworks
PHP	Moderate	No	Large/Active	Yes
Python	Mild	Yes	Active/Growing	Yes

Table 6: Server-Side Language Comparison

3.3.3.2 Frameworks

The focus here was kept to only open-source frameworks, but it appears that the more popular ones are open-source regardless. Three of the more popular frameworks will be explored below.

- **CakePHP**

  This is an easy to implement PHP framework that handles the details from a user's initial request, all the way to the final rendering of the web page. The vanilla version of CakePHP supports database access, validation and caching, while there also exists a number of libraries to implement other functionalities. Helps to maintain application-wide consistency and has built-in security features help to protect the database from attacks.

  This would not be a bad PHP option for our project.

- **Django**

  An everything-included framework in the Python language that includes all the bells and whistles. Django boasts some of the quickest development times, and maintains a dedicated and enthusiastic community. If there is a functionality that is needed, it probably already exists - and without needing to download anything more. This framework helps developers to avoid common security mistakes, and claims to scale well.

  This framework might be more than is needed for our task at hand.

- **Pyramid**

  A lightweight, Python-based framework that is meant to scale well. It's minimalistic design helps to keep things simple, and yet it is extendable with numerous add-ons and packages. The idea here is to start small, with minimal functionalities, and continue to add to it as the needs arise - without having to worry about re-implementing or re-working your code to accommodate.

  This framework's ability to scale may come in handy with our first web application as our needs might not be fully anticipated from the onset.

Framework	Language	Requirements Met	Package/ Libraries Needed	Easy to Implement	Scalable	Secure
CakePHP	PHP	Yes	No	Yes	Yes	Yes
Django	Python	Yes	No	Yes	Yes	Yes
Pyramid	Python	Yes	Yes	Yes	Yes	Yes

Table 7: Server-Side Frameworks Comparison

3.3.3.3 Test Script Generation

Our client's system is currently designed to handle a certain format of a Python script with a .py file extension. As the simulator is already expecting this format, it would be best to recreate it exactly, if possible.

For the purpose of automated script generation, one or more of the generic script templates will be made use of. The design of these templates will become more apparent as progress is made in the project, but repurposing of the templates that our client already has in place should be a possibility. This will help to guarantee an exact match to what is being expected by the simulator.

Creating and storing these scripts from and to the database should be a simple implementation to add to the server-side code in response to user input.

3.3.3 Conclusions: Web Based Application

While both AngularJS and EmberJS are very capable front-end frameworks, **AngularJS** will provide everything that is needed to develop the client side of the application. The lower learning curve in comparison to EmberJS will allow our team to get up to speed quickly and begin developing a prototype. The large amount of documentation available will make it easier to solve issues that may arise when creating the client side application.

When it comes to the back-end development, the decision has been made to stick with **Python** as the language of development. Since most of the frameworks seem to offer what our solution needs and are actually very comparable, implementing our solution with a language that the client is already familiar with just makes sense for future maintenance.

Between the two Python frameworks, Django and Pyramid, **Django** will be implemented. Though there are many functionalities of Django that may never be implemented, it makes sense for first-time developers to have easy access to as many of the common ones as possible, helping to streamline the process as well as decrease development time and effort. With Django's extensive community and tutorial opportunities, it should not take much to start developing with this framework fairly quickly.

For the automated test scripting, **reimplementation of the templates that our client already has in place should be possible**. If this is not the case, great care will need to be taken in our recreations to ensure proper communication with the simulator.

3.3.4 Solution Feasibility and Testing: Web Based Application

A basic application will be built implementing the AngularJS and Django frameworks that can then be used to test for proper correlation. At this point, it must be ensured that the proper requests are sent from the client to the server application. Once the requests are being received by the server framework correctly, testing can begin on the server side application for proper response. This will be done by creating basic hard coded values to return to the client.

Once these tests have passed and our database has been established, testing for correct communication with the database will be done. To do so, previously hard-coded values in the back end will be replaced with server requests, and a client-side request will be made to verify that the expected values have been returned from the database.

For the automated scripting templates, that functionality of our application can simply be run with certain values and it can then be confirmed that the outputted file contains the expected values. In order to confirm that these scripts will be readable by the simulator, however, a trip to the Honeywell offices may be needed to test them live on the simulator, or scripts may be sent to our client to run on the simulator so that adjustments may be made as needed.

Our application will also be tested from a number of popular web browsers to confirm consistency of design and function across the board. The use of a tool such as **Browserstack** may be useful to accomplish this.

4. Integration

In order for our system to function properly, all subsystems need to be well integrated. Given the nature of the product proposition, much of this will be addressed with the frameworks that the project application will implement. The network protocols that will be implemented will be taking care of the rest. These will allow the server-side of our application to communicate effectively with the simulator software and equipment.

Below is a diagram of our system architecture:

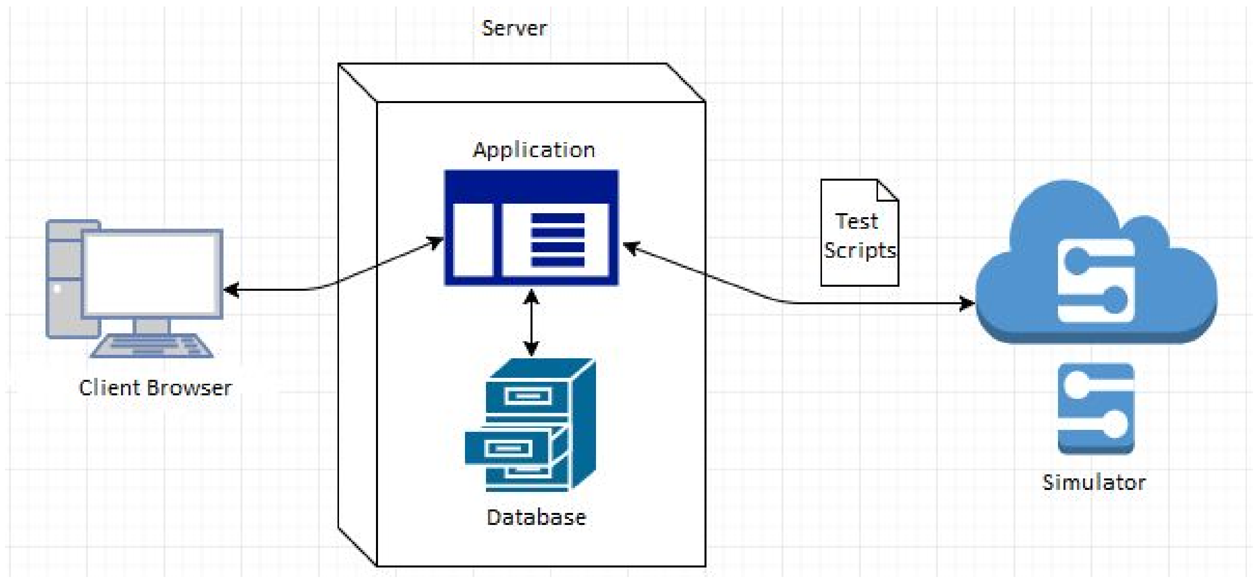

Image 1: System Architecture

As the Django framework has the tools already implemented, the communications between application and client as well as between application and database will be efficiently handled through it. The back-end Django framework and the front-end AngularJS framework are already created with integration in mind.

On the other end, the HTTP protocol to be implemented allows the server to integrate with the simulator by allowing them to communicate. This is essential for making sure test scripts can reach the simulator to be run and so the results of the tests can be returned to the server to be displayed on the client side of the web app.

5. Conclusion

In order for the solution to fit the requirements given by our client for this Next Generation BTU, implementation will involve a number of technologies that must be carefully weighed. Determining the database that best suits the client’s needs, the networking protocols that will best the fit file transfer needs, and deciding on the frameworks that will be used on the web-based application were our main objectives here.

There is confidence among the members of our team that the different approaches being taken to accomplish these challenges are appropriate for the client’s needs. Each seems to fit the challenge’s requirements in order to succeed in the project by choosing the most appropriate Networking Protocols, well-rounded database, and appropriate Front-end and server side frameworks. Should there be exceptions to be made, such as the client looking for a different database based on a different factor or requesting a specific framework, adaptations will be made to assist our client in their goals as best as possible. These changes will be explored through discussion based on what the client is looking for. Furthermore, optimism exists in our team for our current analysis.

For each Technological Challenge previously addressed, the table on the next page summarizes our final proposed solution to said challenge.

Tech Challenge	Handled by:	How it is handled:
Be accessible to engineers regardless of operating system	Application	A platform independent system can be accomplished by creating a web based application that will run equally on all systems.
Be able to store relevant engine simulator tests	Database and Application	When user saves test from front end: ● Frontend sends request to backend ● Backend sends test to database ● Database stores test
Allow engineers access to engine simulator tests	Database and Application	When user requests test from front end: ● Frontend relays request to backend ● Backend makes request from database ● Database returns test ● Backend compiles and returns test to front end ● Front end displays information to user
Have a system that generates performance reports	Database and Application	When a test is completed: ● Application receives results from simulator ● Test report is generated in application ● Report data is sent to database ● Database stores Report
Implement networking to transfer files	Network	When the user runs a test: ● Backend generates test script ● Script is transferred over HTTP to to simulator ● Script is run by the simulator ● Results are transferred over HTTP to the backend
Have the ability to standardize scripts	Application	When a test needs to be run: ● Backend requests test from the database ● Backend produces scripts* based on the test *For a given test the produced scripts are always the same
Have a database for the stored tests	Database	When there is data/tests to be stored: ● Created database using MySQL and structured it to store tests

Table 8: Technological Challenges Addressed