Student Performance Tracker

A Capstone Report
presented to the faculty of the
School of Engineering and Applied Science
University of Virginia

by

Charles Baily

with

Luke Masters
Shivam Mehta
Daniel Seymour
Aiden Smith
Rice Tyler

March 31, 2019

On my honor as a University student, I have neither given nor received unauthorized aid on this assignment as defined by the Honor Guidelines for Thesis-Related Assignments.

Signed: _____

Approved: _____ Date _____
Professor Ahmed Ibrahim, Department of Computer Science

Student Performance Tracker

A Capstone Report
presented to the faculty of the
School of Engineering and Applied Science
University of Virginia

by

Luke Masters

with

Charles Baily
Shivam Mehta
Daniel Seymour
Aiden Smith
Rice Tyler

March 31, 2019

On my honor as a University student, I have neither given nor received unauthorized aid on this assignment as defined by the Honor Guidelines for Thesis-Related Assignments.

Signed: _____

Approved: _____ Date _____
Professor Ahmed Ibrahim, Department of Computer Science

Student Performance Tracker

A Technical Report submitted to the Department of Computer Science

Presented to the Faculty of the School of Engineering and Applied Science
University of Virginia • Charlottesville, Virginia

In Partial Fulfillment of the Requirements for the Degree
Bachelor of Science, School of Engineering

Shivam Mehta
Spring, 2019

Technical Project Team Members
Chad Bailey
Luke Masters
Dan Seymour
Aiden Smith
Rice Tyler

On my honor as a University Student, I have neither given nor received unauthorized aid on this assignment as defined by the Honor Guidelines for Thesis-Related Assignments

Signature _____ Date _____
Shivam Mehta

Approved _____ Date _____
Ahmed Ibrahim, Department of Computer Science

Student Performance Tracker

A Capstone Report
presented to the faculty of the
School of Engineering and Applied Science
University of Virginia

by

Aiden Smith

with

Luke Masters
Shivam Mehta
Daniel Seymour
Rice Tyler
Charles Baily

March 31, 2019

On my honor as a University student, I have neither given nor received unauthorized aid on this assignment as defined by the Honor Guidelines for Thesis-Related Assignments.

Signed: _____

Approved: _____ Date _____
Professor Ahmed Ibrahim, Department of Computer Science

Student Performance Tracker

A Capstone Report
presented to the faculty of the
School of Engineering and Applied Science
University of Virginia

by

Daniel Seymour
with

Charles Baily
Luke Masters
Shivam Mehta
Aiden Smith
Rice Tyler

March 31, 2019

On my honor as a University student, I have neither given nor received unauthorized aid on this assignment as defined by the Honor Guidelines for Thesis-Related Assignments.

Signed: _____

Approved: _____ Date _____
Professor Ahmed Ibrahim, Department of Computer Science

Student Performance Tracker

A Capstone Report
presented to the faculty of the
School of Engineering and Applied Science
University of Virginia

by

Rice Tyler

with

Charles Baily
Luke Masters
Shivam Mehta
Daniel Seymour
Aiden Smith

March 31, 2019

On my honor as a University student, I have neither given nor received unauthorized aid on this assignment as defined by the Honor Guidelines for Thesis-Related Assignments.

Signed: _____

Approved: _____ Date _____
Professor Ahmed Ibrahim, Department of Computer Science

Student Performance Tracker

Chad Bailey
University of Virginia
ceb4aq@virginia.edu

Dan Seymour
University of Virginia
das5fa@virginia.edu

Luke Masters
University of Virginia
lsm5fm@virginia.edu

Aiden Smith
University of Virginia
abs4cr@virginia.edu

Shivam Mehta
University of Virginia
smm2zr@virginia.edu

Rice Tyler
University of Virginia
rht6nf@virginia.edu

Abstract

This project aims to address a number of systemic issues in grading technologies. Current systems fail to give instructors insight into the effectiveness of their teaching, and restricts them to traditional grading structures. It they also cannot advise students on their progress and encourages them to know course material only for the duration of an exam. Student Performance Tracker revolves around classes represented as webs of related topics. Each topic bubble will be assigned a grade based on assignments, related test questions, and grades on prerequisite topics. The (overall) course grade (for each student) will then be an aggregate of the their individual topic grades. The visual aspect aims to allow students to easily identify topics in need of improvement and allows instructors to see the progress of individual students as well as the whole class in order to assess instructional success for individual topics and the class as a whole.

1. Introduction

Our technical project was in support of Dr. Mark Floryan and his goal to improve education. Dr. Floryan is a researcher at the University of Virginia (UVA) focusing on education, with an emphasis on making education better through gamification. Dr. Floryan is a big part of the computer science community at UVA. He not only regularly teaches several courses in the CS department, but he has been
an active member in the CS community at UVA with his involvement in the Association for Computing Machinery and the International Collegiate Programming Contest club.

Dr. Floryan teaches Data Structures, one of the most important courses in the CS department at the University of Virginia. He has had trouble improving the course over the years, as there is a large amount of material covered in it, and a myriad of assessment methods used throughout the course.

The level of complexity of the course obfuscates the actual learning outcomes for students. Dr. Floryan has had difficulty in understanding what his students know by the end of the class and to what degree they comprehend it.

While the normal assessment methods such as tests, homework, labs, and lab reports provide a good level of understanding of how well a student is able to complete an assignment, these assessment methods lack the level of fine-grain detail that would allow the professor to understand how well students understand particular topics as they are assessed and re-assessed over time. Not only are the current assessment methods ineffective at providing detailed information about individual's proficiency in relation to each course topic, but they also take a significant amount of time to implement. The system our team has implemented will allow for a lot of this fine-grained information to be combed from the already existing system with little time commitment. It will also allow for historical data to be used to provide an understanding of how course changes impacted the students learning.

2. Background

To complete our capstone project, our group decided to use two main languages and frameworks. For our backend development, we used the Django framework, which is written using Python. For our frontend development, we use Vue, which is written using JavaScript. Using two different frameworks is a feature that made our capstone group stand out from others because as per the requirements of the class, we were only required to have one.

Django is a web framework that is written in Python. Django tries to make it as easy as possible to start building web applications as possible. To achieve this,

Django focuses on five main points: being very easy to code in, fully loaded, being secure, being scalable, and being versatile [1]. Each of these five main points is a key reason that Django was chosen for our project. For the purposes of this capstone project, Django will be used as an API service. This means that we will not be using Django's built in HTML templates to build a frontend. This is possible due to the flexibility Django. One of the largest reasons that Django was chosen for this project was for the ease of use. For some members of the team, it is their first time in full stack development, therefore a framework with a low barrier to entry was needed. Django solves this by using Python, a very easy to learn programming language that focuses on being human readable [8]. Most programmers have come into contact with Python as it only requires basic programming knowledge. This ease of use is one of the reasons there are so many packages available for python, thus making it easier to build software from open-source resources.

Vue is a progressive framework that is focused on building interactive web pages with JavaScript. Vue combines three different languages into one file: HTML, CSS, and JavaScript. Vue focuses on being the JavaScript framework to go to on smaller to medium scale applications. For the purposes of our capstone project, we are using Vue as the frontend layer. This is a typical usage for the framework and is widely supported. One of the key reasons that Vue was chosen was the simplistic nature of the framework and the power of JavaScript. As mentioned before, as some of the members were not experienced with web frameworks before, it was very important to choose frameworks that were easy to learn. One of the most powerful features of Vue is its component based architecture. This allows developers to build independent sections of code that can be used in many different places. This feature helps to create a uniform experience and save time during development [9]. JavaScript is one of the most widely used web programming languages in the world today. It is used in just about every modern website and web browsers have their own dedicated JavaScript engines [7].

Because our capstone's customer is Dr. Mark Floryan, a professor at the University of Virginia, our production hosting service will be the UVA CS server Paris. Paris is an Ubuntu server running Ubuntu 16.04.6 LTS. In the production environment, Paris is running a cluster of docker containers that scale for the need of the application. There are currently no plans for a domain for the project and is being accessed via an internal IP address.

3. Related Work

The Student Performance Tracker (STP) is a self-paced, project-based learning- a teaching method which utilizes active learning (a teaching method centered around learning through engagement) using projects and tasks where students learn by continuously working with similar, related knowledge on a single project. In the system, computer science students at the University of Virginia are able to go through a course topic by topic. If the student displays and demonstrates mastery of a topic in the course, the student is allowed to proceed to the subsequent topic. However, if the student shows that he or she needs further time learning a subject, then the Student Performance Tracker prompts the user to take ensuing assignments or assessment on the topic.

Furthermore, in this system, students would not be tested on material which they have not learned or covered yet. This custom paced idea of testing allows students to be better and more appropriately tested on whether they properly understand information which they should know-based upon their demonstrated performance rather than a teacher's general assumption.

The Student Performance idea is a UVA computer science specific system branched from the same vein of thought as several preceding constructs: namely ALEKS, Pearson's MyLab and Mastering, Clearing House's project-based learning research projects, and other project-based-learning-centered, web-based systems.

Firstly, Assessment and Learning in Knowledge Spaces (ALEKS) is a learning tool used by UVA. ALEKS was "developed from research at New York University and the University of California, Irvine, by a team of software engineers, mathematicians, and cognitive scientists"[4] and describes itself as:

“ a Web-based, artificially intelligent assessment and learning system. ALEKS uses adaptive questioning to quickly and accurately determine exactly what a student knows and does not know in a course. ALEKS, then, instructs the student on the topics he or she is most ready to learn. As a student works through a course, ALEKS periodically reassesses the student to ensure that topics learned are also retained. ALEKS courses are very complete in their topic coverage and ALEKS avoids multiple-choice questions.“ [5]

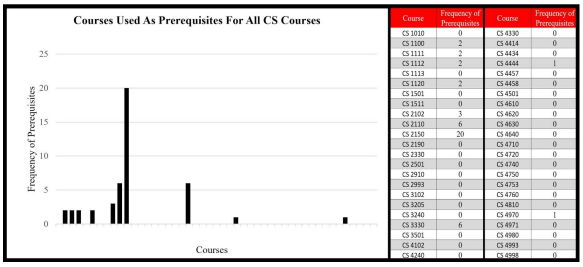
While this tool is useful, part of Professor Floryan's desire for a web-based system is for the system to utilize a multiple-choice bank of questions which he can pull from. Since ALEKS avoids the use of multiple-choice questions, its system would fall short of the desired standard. Furthermore, ALEKS does not have a system made for computer science, it only has a system for math, business, science, and behavioral science [5].

Secondly, Pearson's MyLab and Mastering is another assessment tool used at the University of Virginia. The tool is made for students based on the idea that students learn at various paces. Specifically, the tool provides resources to help "students understand what they know, what they do not, and where to spend their time studying."

Furthermore, assessment is pace-specific; the system has "personalized learning pinpoints the precise areas where each student needs practice, giving all students the support they need \textemdash \space when and where they need it \textemdash \space to be successful" [6]. Unlike ALEKS, Pearson's MyLab and Mastering does have a section for computer science. However, Student Performance Tracker allows complete customizability to UVA computer science professors because the code is given to computer science professors.

Further advantages to using the Student Performance Tracker include eventual UVACollab integration and linking between STP and other websites, external website and resource compilation and collaboration, and a free software tool. With the STP system, communication between UVA's existing systems and STP's system can be uniquely connected through the use of API's; through API's the STP can import and export specific information to other websites. Additionally, many computer science professors use various websites and resources for their subject matter; the Student Performance Tracker system allows professors to have all resources associated with a given topic to students in a single location. Lastly, STP has a further advantage over other software tools because UVA does not have to pay for the use of the system itself.

Currently, Program and Data Representation (CS 2150) is a prerequisite for 20 subsequent courses a student could take after completing CS 2150. Fig. 1 below graphically displays the number of times a course is used as a prerequisite for other courses at the University of Virginia.



Course	Frequency of Prerequisites	Course	Frequency of Prerequisites
CS 1010	0	CS 4330	0
CS 1100	2	CS 4414	0
CS 1111	2	CS 4434	0
CS 1112	2	CS 4444	1
CS 1113	0	CS 4457	0
CS 1120	2	CS 4458	0
CS 1501	0	CS 4501	0
CS 1511	0	CS 4610	0
CS 2102	3	CS 4620	0
CS 2110	6	CS 4630	0
CS 2150	20	CS 4640	0
CS 2190	0	CS 4710	0
CS 2330	0	CS 4720	0
CS 2501	0	CS 4740	0
CS 2910	0	CS 4750	0
CS 2993	0	CS 4753	0
CS 3102	0	CS 4760	0
CS 3205	0	CS 4810	0
CS 3240	0	CS 4970	1
CS 3330	6	CS 4971	0
CS 3501	0	CS 4980	0
CS 4102	0	CS 4993	0
CS 4240	0	CS 6998	0

Fig 1. Prerequisites

Professor Floryan's idea for the Student Performance Tracker came as a result of wanting students to better know information after finishing CS 2150. He observed that students were coming out of the course knowing some information well, but only knowing all of the information students were supposed to know for subsequent courses. To combat this, he wanted to create a way that students would have to go back and learn information they do not adequately know as well as not proceeding through a course to a topic which they are not ready for, based on their demonstration on preceding topics. Fig. 2 shows Professor Floryan's topic-by-topic layout of CS 2150. After mapping out the topics, he came up with the idea of a web-based program that allows students to individually progress through the course on a by-topic basis. Student Performance Tracker is a tool that directly addresses the specific need of Professor Floryan [2].

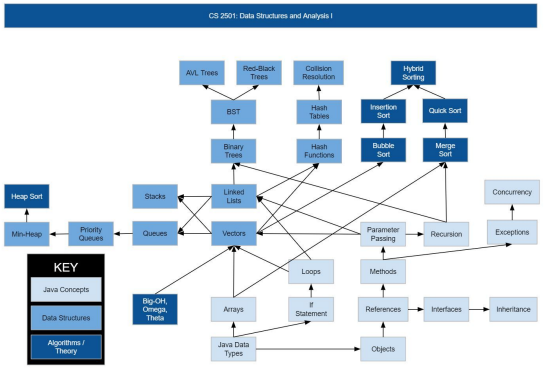


Fig 2. CS 2150 Class Graph

3. System Design

3.1 High Level Description

The general purpose of our system is to create a different way of progressing through the content of a class. This is done by conceptually breaking down classes into a set of related topics that precede and follow one another. Thus, the centerpiece for each class on the website is the graph that represents the sum of its content.

There are two key types of users in our system: professors and students. Professors are responsible for creating courses, structuring their content, grading, and enrolling students. Students are responsible for traversing the graph and learning the content of the class. Both use the same underlying model in the database, but are differentiated based on their command over different class objects, which only give permission to certain users to edit them.

Professors can do a myriad of things within the system. Once a professor signs up, they first need to create a class. Once they create a class, they can create and edit the topic graph for the class at any time. Once the graph has been created, they can create quizzes for any of the topics or add information resources for those topics. They also have the ability to enroll students into their courses, upload grades for, and view the progress of any of their students for that course. If they choose, they can also delete classes. Professors will eventually also have the ability to import grades from external services as well to update grades internal to the system.

A student user has limited capabilities in comparison to their professors. They are enrolled in classes

by their professors and cannot create classes of their own. They can only view and update their own progress on the quizzes and assignments by taking them. They can take quizzes as frequently as they would like, but they cannot create their own. Their key focus in using the system is traversing and filling out the graph, so they only need to focus on their own material.

To separate functionalities between students and teachers, we took two levels of protection. First, all elements relating to professor specific functionality were hidden to student users using conditional rendering and a global variable determining the role of the user. Those requests still would not work if users could access them because the requests were protected on the backend using a mix of role checking (i.e. only professors can make certain requests) and Google's OAuth system determining exactly who can do what.

3.2 Overall system architecture

The system is composed of a three tier architecture made up of a frontend, backend, and database. The frontend is the website users see and interact with. Their interactions then send requests to the backend. When the backend receives those requests, it interprets them, processes some logic, and interacts with information from the database. When the backend is done processing that logic, it sends back some data to the frontend so it can update the website with new information.

3.2.1 Frontend

This is the website that users access. It is served from a separate port from the backend. We created a Javascript single page web app that sends the entire app as a single package to users when the website is requested. The Javascript app processes all navigation, rendering, and event responses with no need to interact directly with the backend.

3.2.2 Backend

Our backend is a Django REST API. It acts as the middle ground between our frontend and our database. It takes requests from our frontend which involves retrieving and updating data. The backend then interprets those requests and retrieves the necessary data from the database.

3.2.3 Database

The database is the most simple of the three, as Django's model framework obscures the language specific aspects of whatever database architecture you are using. We used a Postgres database, but as mentioned previously, it had no major bearing on our development.

3.3 Design Decisions

This single page web app + REST backend architecture is very common in modern companies' tech stacks and provides a good entry point into the modern work world. Despite this, it is fairly complicated and not taught as the core curriculum at UVA. Thus, it was a very large learning curve for some of the team. We mitigated this by having office hours held by the team members with more experience in this kind of tech stack, pair programming, and using Slack frequently.

Creating a separated three tier architecture as opposed to using Django's built in full stack suite was a design decision in itself. The decision to choose this (admittedly more difficult) architecture was two fold. First, as previously mentioned this is a very common architectural pattern used in modern technology so it provides good practice/learning opportunities for the team. Second, using a containerized/compartmentalized architecture allows for better separation between functionality and minimizes code bloat.

We decided to use SemanticUI for our frontend component library with some scant custom CSS instead of using all our own CSS for several reasons. The first was that very often companies will either (a) have their own in house component libraries to keep a consistent design language across products or (b) use a third party component library like SemanticUI or Bootstrap to speed up development time. Understanding vanilla CSS is an important building block of knowledge, but a key goal was integrating and experiencing work similarly to an actual work environment. SemanticUI also has a fair amount of components that we knew we would need for our application and had a well implemented Vue fork, so we chose it over other similar options like Bootstrap.

Another challenge was incorporating Google OAuth into the tech stack. Our client required using Google single sign on as the only method of logging on, so we could not use Django's built in authentication system. Understanding where it stood in between the frontend and backend was a non-nominal issue that took some figuring out. Originally, the frontend made the request straight to Google, then sent the result to the backend, but after being advised that this was unsafe, the feature was rebuilt so all the functionality was handled in the backend. Because our architecture relies on asynchronous requests sent from the frontend to the backend, ordering our requests and understanding asynchronous operations became an important challenge to overcome. There are a few cases on the frontend where there are deeply nested asynchronous requests that make processing the logic difficult, as such if we were to have more time, we would move more of that functionality to the backend to make synchronous processing more readable.

3.4 Graph

Our graph took some careful design to structure it properly. As a 2D graph, it can be described as a system of nodes and

edges, where the Topic model is the node and our TopicToTopic model is the edge between the nodes. The Topic model describes what the topic is about and what class it belongs to and the TopicToTopic describes the direction of the edge, meaning which topic you have to complete before moving onto the next topic. This graph is further complicated by students all having their own progress through the graphs, so we needed a model, StudentToTopic, to track all the students progress in an individual topic. As the project went forth we knew that we needed the progress for a specific topic to potentially come from multiple sources (i.e. a homework and a quiz), so we need a Grade object to reflect that that could be tied to a StudentToTopic.

Because the keystone of our application is the directed acyclic graph that represents a class, getting it visually right was important. Our client had given the idea significant forethought, so taking his direction to get the structure of the graph was easy enough. However, actually implementing the graph was difficult. This is because the graphing library we were using was a fork of an original graphing library suited to the frontend framework we using. This fork was an incomplete copy of the original library, so there were some cases where we had to extend it with our own functionality.

3.5 CSV Upload and Grading

One of our client's key requests was to be able to create courses and grades by importing CSV files. Thus, we had to figure out how to structure those CSVs keeping in mind the topic/graph construction of the classes in our website as well as how the database was structured. When we brought a finished CSV upload feature to our client, he did not like the structure we had generated for it, so we had to go back and restructure our CSV system. This is due to our team not accounting for the real-life use cases of the system, such as needing to easily and quickly convert TPEG grade documents into student grades.

Another key requirement was for the application to be able to consume data from other external services and incorporate it as a part of the core service. For example, being able to import grades from UVACollab to update grades on the service. After some thorough research on UVACollab and realizing that would not be possible, we set out to create a test external site, separate from our entire application, where users could take quizzes and those grades would feed into the main site as a proof of concept.

Originally the only grades being stored in the site were for the topic quizzes which were developed in the Fall semester of 2018. Because they were the only items contributing towards grades originally, we did not need to build an extra system to support multiple types of grades. This changed when we realized we would need to import grades from homework and tests. As such, in early Spring we overhauled the way we calculated grades for all students that involved storing grades for quizzes, tests, homework, and custom categories made by professors.

4. Results

Although our system is not currently in use, we will cover the future impacts of our system as it is currently implemented. Our website will improve the relationship that students have with the course material as well as the relationship that teachers have with their courses.

The previous system obfuscated the skill level of each student in relation to individual topics. This reduced the professors ability to improve the course based on students' results. Our site allows for teachers to implement a unique grading system on top of their already existing course which makes implementation of the system extremely simple. The original system is made up of graded homework, tests, and quizzes which represent a variety of course topics. These are individually graded with an overall grade assigned to each topic, each of which has some notes as to what the student could improve. This methodology requires students to be self-reflective and focused on learning in order to understand exactly how well they understood the concepts which were covered in each assessment.

Our system allows for teachers and their assistants to provide a mental map for students as well as other functionality with little effort. Initially, a professor will have to compile a list of course topics and figure out what kinds of relations the topics have with one another. They will then upload these course topics along with associated grade categories to create the initial course mapping. Then as assignments are graded, there will be additional grading criteria for each question as to how much they each relate to each topic which the questions tests. This is all made to be as easy as possible with basic spreadsheet support. The system we are creating will not only provide better associations between a students work and their understanding of the material, but it will also allow a teacher to visualize how changes to their courses impacts student learning through the overall grades of students for each topic in each semester. This is in addition to the ability for teachers to use questions from previous semesters as quiz questions to allow students to easily review material. Overall our system is low-impact for the course administrators with a high impact on both student and professor.

5. Conclusions

Student Performance Tracker is a University of Virginia self-paced teaching tool for computer science classes which not only allows students to work at their own pace, but also

evaluate their performance on a topic by topic basis. Additionally, professors can extract data from individual students and/or the whole course, which can be used to improve future classes and students. Student Performance Tracker is not the only software out there that does these basic evaluations for students and professors. ALEKS, Pearson's MyLab and Mastering, and Clearing House are almost equivalent to Student Performance Tracker. Some of these related tools do not have computer science as a subject to study, while others are not as customizable. The most advantageous part of STP is not only its ability to pull information from other resources and websites and utilize them in one location, but also the visuals it provides. By making a myriad of design decisions, like choosing Django REST framework, Semantic UI, and Google OAuth, we were able to create a highly customizable, visually adaptive learning environment. Using graphs to help students identify their progress and CSV uploading for Professor Floryan's convenience, this tool will eventually exploit the topics that are not being fully covered.

Overall, Student Performance Tracker was developed to enhance the learning experience for students by accurately gauging their aptitude on a topic by topic basis rather than assessing students on a variety of topics, which does not completely represent the students knowledge.

6. Future work

Professor Floryan has said that this will be an ongoing project that will eventually be used in classes. His plan is to play with existing data this summer to figure out kinks and confirm suspicions he has had with course completion and course material competency. This will be used to specifically look at the material being taught in CS2150. He suspects that there is currently not a sufficient amount of testing being done on the course topics. Next year, he plans on handing it over to another team to further develop it.

One area that needs to be developed further is inter-class connection. As it stands, each class exists as an independent graph. Professor Floryan indicated that eventually he wants each class to be a subset of a greater graph that represents either the department or the CS degree requirements as a whole. External site integration is another avenue of interest. The base for supporting external sites and having them update grades internally is there, but the actual integrations (outside of our own proof of concept external site) are not there yet. Additionally, Professor Floryan indicated an interest in expanding the project to include a number of accessibility features for the differently abled including accessibility for blind users.

Acknowledgments

This project was completed with the help of Mark Floryan who contributed the original project idea, and helped us refine our implementation for wide distribution.

References

[1] Django Software Foundation. [n. d.]. Django Project. Retrieved March 31, 2019 from https://www.djangoproject.com/

[2] Mark Floryan. [n. d.]. Mark-richard-floryan. Retrieved March 31, 2019 from https://engineering.virginia.edu/faculty/mark-richard-floryan

[3] McGrawHill. [n. d.]. Course Products. Retrieved March 24, 2019 from https://www.aleks.com/about_aleks/course_products

[4] McGrawHill. [n. d.]. Overview of ALEKS. Retrieved March 24, 2019 from https://www.aleks.com/about_aleks/overview

[5] McGrawHill. [n. d.]. What is ALEKS. Retrieved March 24, 2019 from https://www.aleks.com/about_aleks

[6] Pearson. [n. d.]. MyLab and Mastering allow you to. Retrieved March 24,2019 from https://www.pearsonmylabandmastering.com/northamerica/educators/features/index.html

[7] Pluralsight. [n. d.]. JavaScript. Retrieved March 31, 2019 from https://www.javascript.com

[8] Python Software Foundation. [n. d.]. Python Project. Retrieved March 31, 2019 from https://www.python.org/

[9] Vue.js. [n. d.]. The Progressive JavaScript Framework. Retrieved March 31, 2019 from https://vuejs.org/v2/guide/, Vol. 1, No. 1, Article . Publication date: April 2019.