EPiC Series in Computing

Volume 77, 2021, Pages 70–79

Proceedings of ISCA 30th International Conference on Software Engineering and Data Engineering

VS-TAP: Veteran Services Tracking and Analytics Program

Jonathon Hewitt, Daniel Hall, Christopher Parks, Payton Knoch,
Sergiu Dascalu, Devrin Lee, Nikkolas J. Irwin, and Frederick C. Harris, Jr.

Department of Computer Science and Engineering,
University of Nevada, Reno, NV USA
fred.harris@cse.unr.edu

Abstract

The Veteran Services Tracking and Analytics Program (VS-TAP) is a web application used to store and query the rate and duration of visitors within Veteran Services' locations. The application accepts data from Navigate as well as a hosted demographics survey to display statistics in a graphically meaningful way. Accumulating data from different sources allows stakeholders to create custom reports to compare multiple variables that represent student veterans.

Keywords: Analytics, Authentication, Data, Database, Django, Document processing, ETL (Extract, Transform, Load), Systemd-nspawn, Tracking, Veteran Services, Visualization, Web application

1 Introduction

The Veteran Services Tracking and Analytics Program (VS-TAP) is a data gathering and analytics application. The goal of this program is to collect, store, and combine data from several sources into a single usable database. The web application tracks the rate and duration of visitors that attend veteran centers and events. The program also combines all the data collected from various sources that can be queried for data visualization purposes. Data capture and visualization are important to the center's existence and helps determine the success of events as well as requests for funding.

The interface for data visualizations is presented as a "reports wizard" to help walk Veteran Services staff through graph generation. The initial aim was to mimic the quantity of graphs associated with Microsoft excel while eliminating the learning curve. The reports wizard was designed to give staff members more control over graph axis, titles, and graph aesthetics than was previously possible using Microsoft excel.

Concerning security, VS-TAP was be designed to protect against malicious actors. To this extent developers integrated user authentication, protection from SQL injections to the database, as well as CSRF (Cross-Site Request Forgery) token validation. In addition to the security mentioned, VS-TAP is only accessible from the University of Nevada, Reno (UNR) network to limit external network traffic.

The VS-TAP web application was designed to be containerized using systemd-nspawn which is native to the Linux operating system. In May 2021, VS-TAP was launched on the Department of Engineering's virtual server at the University of Nevada, Reno.

The rest of this paper is structured as follows: Section 2 presents the motivation and design of VS-TAP including Functional and Non-Functional requirements as well as the application's Use Cases. Section 3 covers the technologies used to implement the current version of VS-TAP. The final version of VS-TAP along with screen shots are given in Section 4. VS-TAP conclusion as well as future works are given in Section 5.

2 Motivation and Design

Manually collecting visit data is difficult and unreliable, and the kinds of reports you can generate from this data is limited. By automating the check-in and check-out procedures at the Veteran Services offices and collecting data in the process, the amount of useful reports that can be created increases. The main goals for this project is to provide a seamless check-in and check-out experience and to augment the kinds of reports that can be generated. To make sure these goals are adequately met, a list of functional and non-functional requirements are created alongside a list of desired use cases.

2.1 Functional Requirements

Functional requirements, per Ian Sommerville [5], are used to describe the necessary functionality of a system. These requirements are directly seen in the final project. The following is a list of functional requirements for the VS-TAP system.

1. The site will be hosted and run on the UNR network.
2. Parse scanner data from Navigate.
3. Store visit and demographic data in a database.
4. Allow users to query the database for data reports and display on the reports page.
5. Allow users to specify events for visit data.
6. Allow users to create an account.
7. Allow users to log in to their accounts.
8. Implement a navigation page that links each page on the site.
9. Allow users to export reports as images for reports.
10. Allow users to export report tables as CSV files.
11. Allow for users to search individual students.
12. Allow for users to remove individual students from the visit data.
13. Display different home pages for authorized and unauthorized visitors.
14. Provide a wizard as a user interface for creating new reports.
15. Allow users to specify a range of dates for reporting.
16. Allow users to change their password.
17. Allow users to upload a profile picture associated with their account.
18. Allow users to change their account first and last name.
19. Allow users to change their email address.
20. Support manual upload of visits when scanners are unavailable.
21. Allow users to quickly query for individual statistics e.g. average visit duration.

22. Allow users to save templates for data visualizations and load them with new data points.
23. Provide an administrative page for managing all user accounts.
24. Allow administrators to change names, email addresses, passwords, and profile pictures of other users within the system.
25. Allow users to change the name of each saved report type.
26. Provide a dynamic wizard page for adding stacked graphs.
27. Allow users to download reports as PDF files.
28. In addition to the wizard, provide an interactive dashboard for quickly creating new reports.
29. Automatically import visit data from Navigate on a live basis.
30. Provide a portal for quickly sharing visit data to other users.

2.2 Non-Functional Requirements

Non-functional requirements, per Ian Sommerville [5], are used to describe the quality constraints that a system must satisfy. The following is a list of non-functional requirements for the VS-TAP system.

1. Allow for multiple concurrent users to upload data and create visualizations
2. The site should return queries for data, and data visualizations quickly
3. The site should be easy to navigate for people with little to no technical knowledge
4. The data reporting options should be shown in a straightforward and usable manner
5. The site should have minimal downtime
6. The site should be robust to bad data uploads
7. The site should be non portable and only accessible from the campus network
8. Users should be able to obtain all visual reports that are needed for funding of VS
9. All information protected by FERPA must be secure from unauthorized access
10. The software should be designed in a way that does not need frequent updates
11. The code should be easily maintainable in case future updates to the software are necessary
12. The software should function offline during downtime

2.3 Detailed Use Cases

- **AccountLogin**
  When the user first enters the website, the user will be prompted for a user name and password. If the credentials are correct, the user will be taken to the home page.
- **ChangePassword**
  If the user wants to change their password, they can select `Change Password`. The user will be prompted for their current password, the new password, and a second entry for their new password. The current password must be correct and the two entries for the two password must match. If all forms are correct, the user will receive a message that their password was successfully changed. If the current password is incorrect or the two fields for the new password do not match, an error message will display.
- **SelectReportPage**
  When the user selects `Visualizations` from the navigation bar or enters the "Visualizations" view from the address bar, the user will be taken to the visualizations page.

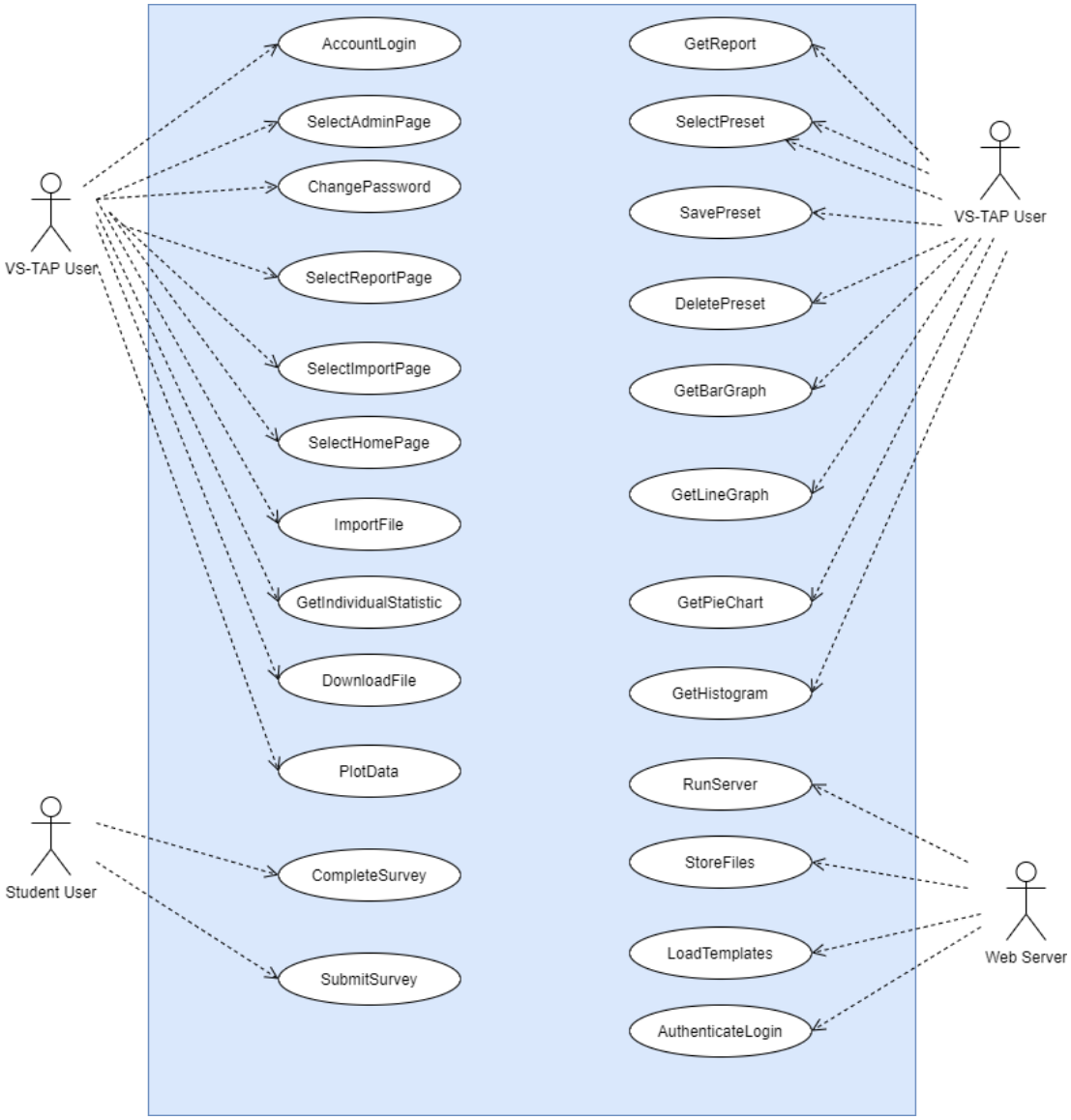

Figure 1: Use Case Diagram

- **SelectImportPage**
  When the user selects `Upload Files` from the navigation bar or enters the "Import" view from the address bar, the user will be taken to the upload page.
- **SelectHomePage**
  When the user selects `Home` from the navigation bar or enters the "Home" view from the address bar, the user will be taken to the home page.
- **ImportFile**
  On the **Import** page, the system will prompt the user for a file. The user will select the file from their computer. If the file is successfully uploaded to the server, the system will indicate to the user that it was successful; otherwise an error message will display. After a successful upload, the parser will begin parsing the file.
- **GetIndividualStatistic**
  On the **Reports** page, one of the options that the system provides to the user is the ability to select an individual statistic (e.g. average G.P.A, total number of visitors on 10/31/2020). The user will select from the available individual statistics, then click `Get Individual Statistic` to obtain the statistical report.
- **DownloadFile**
  On the **Reports** page, the user will have the option to download any data visualization that they select to their computer. For example, if they select a bar graph, they can download the graph as an image.
- **PlotData**
  On the **Reports** page, the user will select from a list of options for a specific type of graph. After selecting the options, the user will click a button that will submit a user request to the system to plot the data. The visualizations module should return the plotted data to the user in the form of a graph.
- **CompleteSurvey**
  Upon the first visit of the VS office, the student is taken to the **Survey** page to complete a list of fields.
- **SubmitSurvey**
  Upon completing the survey, the student clicks `Submit`. If all fields are correctly filled out, the survey data is inserted into the database; otherwise, an error message appears.
- **GetReport**
  After filling out the reports wizard, users can get a detailed report about attendance for a given data range, including both visual and tabular data.
- **SelectPreset**
  After selecting a specific saved report from the list of saved reports, a user is given the details of that report with options, such as creating a new report from the saved report preset and deleting the preset altogether.
- **SavePreset**
  After obtaining a report from the Reports Wizard, the user is given the option to save the preset. If the user saves the preset, the the preset is saved into the database where the use can access it via the **Presets** page.
- **DeletePreset**
  The report preset is deleted from the database after the user selects `Delete Preset` and the preset no longer appears in the list of saved presets.
- **GetBarGraph**
  In the reports wizard, the user obtains visit, demographic, and/or survey data in bar graph format.

- **GetLineGraph**
  In the reports wizard, the user obtains visit, demographic, and/or survey data in line graph format.
- **GetPieChart**
  In the reports wizard, the user obtains visit, demographic, and/or survey data in pie chart format.
- **GetHistogram**
  In the reports wizard, the user obtains visit, demographic, and/or survey data in histogram format.
- **RunServer**
  Upon execution of `manage.py`, the web server loads the software and makes it available to its users.
- **StoreFiles**
  The web server will maintain storage of all files, including database files and user profile pictures.
- **LoadTemplates**
  In conjunction with Django, the web server is responsible for loading all template (HTML) files that will display web page content to the end user.
- **AuthenticateLogin**
  The web server should authenticate the user's credentials when they try to log into the systems. If the password or username are not correct, the server shall deny user access to the system.

3 Technologies Used

VS-TAP uses Django as the main architecture. As VS-TAP is a web application, the frontend features Hypertext Markup Language (HTML), Cascading Stylesheets (CSS) to provide visual enchancements to the object displayed via HTML, and JavaScript to provide any interactivity to the users. The backend logic is handled via Python scripts. SQLite3 is the database containing all of the visit and demographic data. Additionally, the team used the Bootstrap HTML library for faster frontend development time and JQuery for handling user events in JavaScript.

Django: Django uses a concept known as Model-View-Template (MVT), which is based off of the Model-View-Controller architectural pattern [3]. Django models feature objects that interact with the integrated database, such as SQLite3. A model object contains all of the database fields associated with the object. Each instance of the object corresponds to an entry in the database. The View contains all of the functions that render the HTML templates and the associated logic performed prior to the rendering. The Template is the HTML templates that are displayed, including any accompanying JavaScript or CSS styling.

HTML/CSS/JavaScript: A majority of the frontend starts with a base HTML page that contains the common styling and layout used for all of the web pages. Specific web pages (e.g. Reports page) extend from the base HTML page. Django provide dynamic elements in the HTML pages through the use of context variables. A context variable is a variable whose value is calculated by the backend via Python functions. The output varies based on conditions such as the database contents and user input. CSS provides styling to individual HTML elements, classes of HTML elements, or entire pages. CSS style options include (but are not limited to)

centering, changing the color, font size, and font color. JavaScript provides interactivity to the page based on user actions, such as clicking on a button and typing in a text entry.

Dash: Dash is a library used for creating detailed and informative interfaces that provide visual reports through Plotly [2]. VS-TAP uses Dash because each visit report needs visual representation, such as a Bar Graph or a Pie Chart. Django contains an extension known as Django-Plotly-Dash. Django-Plotly-Dash provides tools for integrating Dash components within the Python scripts and HTML code. The backend of the reports page is written using Plotly functions and variables in Python while the graph itself is rendered by Dash.

SQLite3: SQLite3 is used for the database. The database logic for the user accounts and the report presets is automatically carried out by Django models. The logic for the student demographics and visits are direclty handled by SQLite3 commands embedded in the Python view functions within the parser and the reports modules.

4 Results

Aside from security, such as user authentication, there are three main sections of the VS-TAP web application that veteran services' staff and visitors interact with: the student survey, the document upload section, and the data visualization page.

Student Survey: The student survey is a multiple choice questionnaire which is accessible by students using a QR code located within each veteran center. Data collected from the student survey helps veteran services determine relationships between various demographic data, student's involvement in the center, and student's academic performance. Figure 2 and 3 show the beginning and end of an 18 question survey that helps the VS-TAP web application create more dynamic reports to better serve student veterans.

Document Upload: The document upload section allows users to upload documentation from different sources. Once a document is submitted to the application, the web application extracts, transforms, and loads (ETL) the data into the back-end of the application. The document upload section allows the users to upload data from the university's Navigate system, GPA data, and manual entry data in the event that the navigate system is down. Figure 4 show the document upload section where users may upload two different comma separated value (csv) documents or enter manual student's visit data.

Data Visualization: The data visualization section walks the user through a "reports" wizard to create a graphical representation of specific visitor's data. Reports creation allows the user to generate various different graph types while querying 28 different student visitor parameters. Figure 5 and 6 show the results from one such query where the reports wizard creates a table and graph for the classification of students who visit the center between the dates of 04/05/2021-08/31/2021.

5 Conclusion and Future Work

The staff of VS can now use spreadsheets obtained from Navigate to upload them to VS-TAP. By uploading the spreadsheet data, the data is saved to a database and the data storage process

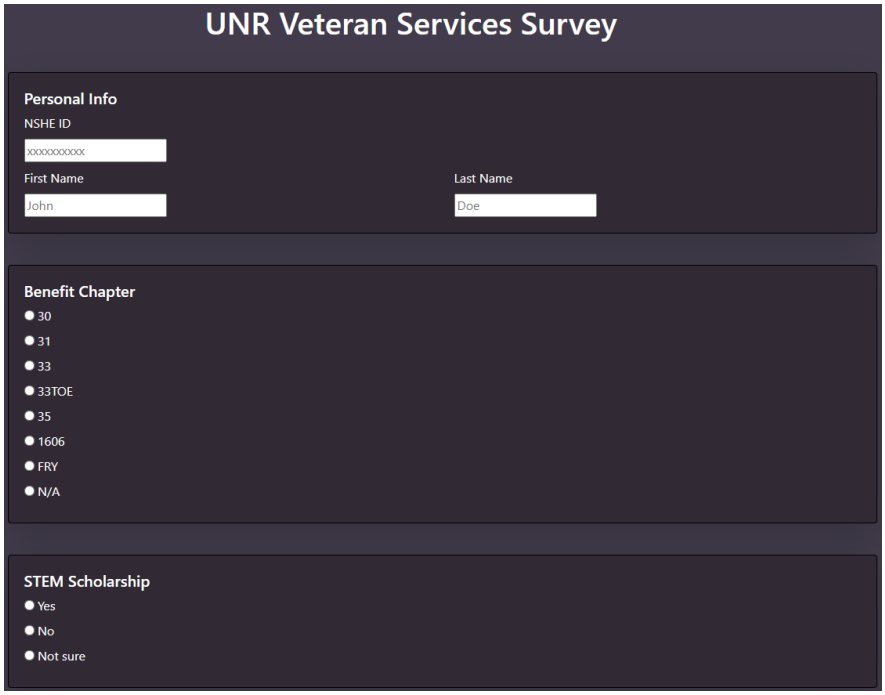

Figure 2: The UNR Veteran Services Survey requires each visitors NSHE ID to relate their survey demographics to their visit data.

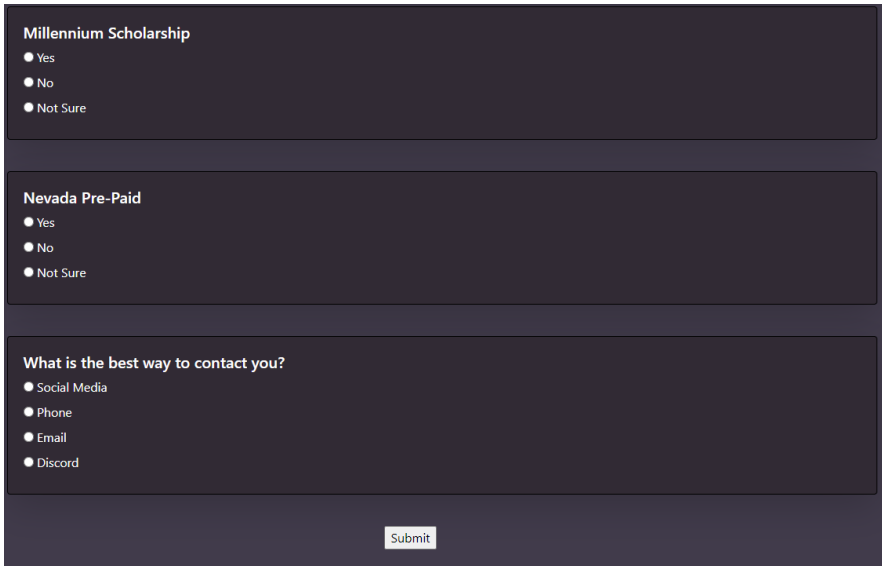

Figure 3: When each survey is submitted the student's responses are stored along their visits data for future querying.

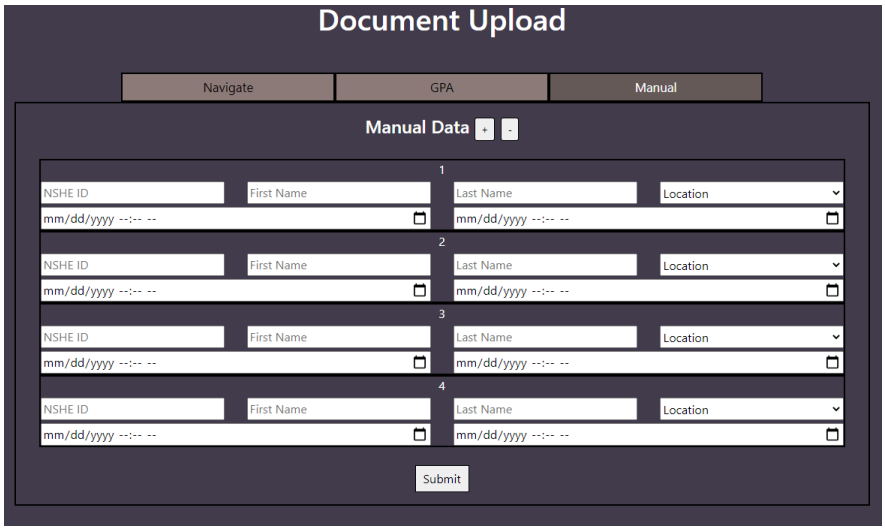

Figure 4: In the event that the university student tracking system is offline, staff at Veteran Services may still upload visits data manually.

Count of Classification, All Locations, from 04/05/2021 to 08/31/2021

Row Labels	Count of Location
Freshman (2020 Fall)	3383
Junior (2020 Fall)	3455
Senior (2020 Fall)	3221
Sophomore (2020 Fall)	2728
Grand Total	12787

Figure 5: Table representing the amount of students who visit the center and their associated classification year.

is automated. Staff members of VS can specify the type of report, date range, and aesthetics to retrieve a report that is automatically generated. Previously, VS staff needed to manually inspect multiple spreadsheet pages to create a custom report for VS funding. VS-TAP has the potential to be used for other buildings at both the UNR campus and other Universities. Other universities provide an equivalent to the Veteran Services building and may use a similar funding structure; therefore, it would be beneficial for other universities to use this software to track attendance.

Although there exists commercial software that tracks attendance, such as ADP [1], the commercial software is primarily concerned with tracking employee attendance for payroll purposes. VS-TAP customizes the attendance tracking to provide data visualizations and reports to obtain funding. In the future, VS-TAP's functionality can also be expanded to a more interactive dashboard of data visualizations. Currently, reports are generated by selecting from a list of customization options through a Wizard format. If users can dynamically view how their customization choices can change the output of their reports, they can find their ideal customization settings at a faster rate. The interactive dashboard can be achieve through libraries

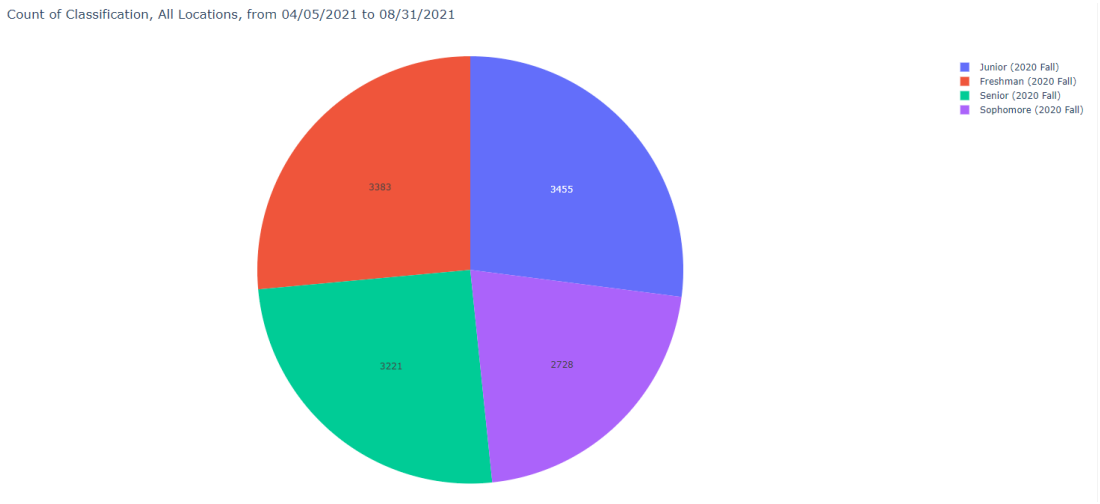

Figure 6: Pie chart representing the amount of students who visit the center and their associated classification year.

such as Dash Enterprise [4].

VS-TAP requires users to upload visit data from a third party source. The third party source is Navigate. In the future, it would be beneficial if visit data reflected in real-time. Real-time visit data can be obtained through an auxiliary application within VS-TAP that uses hardware to scan the WolfCard, then uploads the visit to the VS-TAP database through a cloud infrastructure. By allowing real-time visit uploads, VS staff would be solely focused on obtaining the desired report needed for funding.

Acknowledgments

This material is based in part upon work supported by the National Science Foundation under grant number IIA-1301726. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation.

References

[1] ADP. Employee time tracking. https://www.adp.com/what-we-offer/time-and-attendance/employee-time-tracking.aspx.

[2] Dash. Dash python user guide. https://dash.plotly.com/.

[3] Django Software Foundation. Django documentation, release 3.2.4.dev. https://docs.djangoproject.com/en/3.2/#django-documentation, 2021.

[4] Plotly. Dash overview. https://plotly.com/dash/.

[5] Ian Sommerville. Software engineering 6th edition. https://www.pearson.com/us/higher-education/program/Sommerville-Software-Engineering-6th-Edition/PGM267706.html.