Design and Implementation of VS-TAP The Veteran Services Tracking and Analytics Program

Jonathon Hewitt*, Daniel Hall*, Christopher Parks*, Payton Knoch*,
Sergiu M. Dascalu*, Devrin Lee*, Nikkolas J. Irwin*, Frederick C. Harris, Jr.*
University of Nevada, Reno,
Reno, Nevada, USA.

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.

Key Words: 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 visualization 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. Previously, Veteran Services manually tracked attendance using a USB-connected barcode scanner. Veteran Services staff were unable to obtain demographic information directly from the barcode scanner. After tracking attendance with the barcode scanner during a given time frame, staff members would periodically send the data containing student information to the Office of Data Analytics. The staff at the Office of Data Analytics would match the demographics with the student barcode information and send an Excel report back to Veteran Services. The Excel sheet would display demographic information for each associated student barcode entry.

Veteran Services also collected additional demographics that were not available from the Office of Data Analytics. Veteran Services used a custom Google survey from an iPad device. First-time visitors would fill out the survey on the iPad upon entry into the facility. Staff members would periodically export the survey data via an Excel spreadsheet. Staff members would have a total of three spreadsheets to build reports: Attendance in/out information, demographics provided by the Office of Data Analytics, and the Google survey demographics. Using the three spreadsheets, staff members would manually reconcile and match student data to build the reports using chart wizards provided by Microsoft.

During the development of VS-TAP, VS implemented an upgrade to the barcode scanner system. VS implemented a student identification (WolfCard) scanner. The upgraded scanner is able to scan student ID cards and allow Veteran Services staff direct access to demographic information instead of obtaining this information from the Office of Data Analytics. VS staff continues to use a survey to obtain supplemental demographic information. VS-TAP includes the built-in implementation of the survey that directly feeds survey data into the database, instead of using a Google Forms survey. VS-TAP aims to allow staff members to upload only two spreadsheets that are automatically parsed and updated into the database. The staff can then use a reports wizard to obtain the appropriate charts and tables. 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 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 [5] which is native to the Linux operating system. In May 2021, VS-TAP was launched on the College of Engineering's virtual server at the University of Nevada, Reno.

The rest of this paper is structured as follows: Section 2

*Department of Computer Science and Engineering. 1664 North Virginia Street / MS 171. Reno, NV 89557. Email: {jonathonhewitt, danielhall, christopherparks, pknoch}@nevada.unr.edu, dascalus@cse.unr.edu, dllee@unr.edu, nikkolasjirwin@nevada.unr.edu, fred.harris@cse.unr.edu

presents the motivation and design of VS-TAP including functional and non-functional requirements as well as the application's use cases. Section 4 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 5. VS-TAP conclusion as well as future works are given in Section 6.

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 Similar Applications

Data analytics are commonly used across multiple industries. Tablaeu, part of Salesforce's software suite, allows organizations to analyze and visualize data from multiple sources that is fed into a single platform [10]. For example, users of Tableau can use data from sales, marketing, and business expenses to generate detailed, visual reports [10]. VS-TAP provides a similar concept - using a central location for importing and visualizing data. The difference between VS-TAP and Tableau is that VS-TAP is more specialized for attendance tracking that is build to incorporate the specific third-party technology that is already used at Veteran Services.

Attendance tracking is commonly used among businesses for hourly employees. Kronos is a timekeeping software that is used to track employee attendance, employee time off and vacation, help businesses with remaining compliant with labor regulations, and provide detailed visualizations [4]. Kronos software allows businesses to obtain detailed demographic information based on employee attendance, such as employee count by state, comparing shift hours worked against shift hours scheduled, and employee headcount by business location [3]. Kronos is the most similar software to VS-TAP in that it is primarily used for tracking attendance. Kronos tracks existing employees that are in regular attendance. While VS-TAP has visitors in regular attendance, VS-TAP is meant to handle new visitors on a daily-basis with the integrated survey. Additionally, VS-TAP is meant to provide a lower learning curve for its targeted users.

Microsoft Excel is another tool used for tracking attendance and generating reports. Excel allows users to manually enter data into tabular format, known as a ”spreadsheet” [6]. Prior to the development of VS-TAP, Microsoft Excel was used by staff members. Excel requires its users to manually filter out unnecessary or repetitive data that is not used in the visual reports. Excel also allows its users to build charts by selecting the relevant data entries and choosing from multiple options in a wizard. Additionally, if multiple spreadsheets are used in a single chart, it requires its users to combine the spreadsheets. Filtering out data and combining the spreadsheets can take up to several hours. Although VS-TAP still involves Excel spreadsheets, it automates the data selection for report generation based on user criteria. Furthermore, VS-TAP automates the process of filtering and combining spreadsheets.

2.2 Functional Requirements

Functional requirements, per Ian Sommerville [9], 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.

The System shall:

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

24. Allow users to change the name of each saved report type.
25. Provide a dynamic wizard page for adding stacked graphs.
26. Allow users to download reports as PDF files.
27. In addition to the wizard, provide an interactive dashboard for quickly creating new reports.
28. Automatically import visit data from Navigate on a live basis.
29. Provide a portal for quickly sharing visit data to other users.

2.3 Non-Functional Requirements

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

2.4 Detailed Use Cases

This subsection presents the detailed use cases. Figure 1 gives the use case diagram.

- **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 passwords 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.
- **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 preset is saved into the database where the use can access it via the **Presets** page.

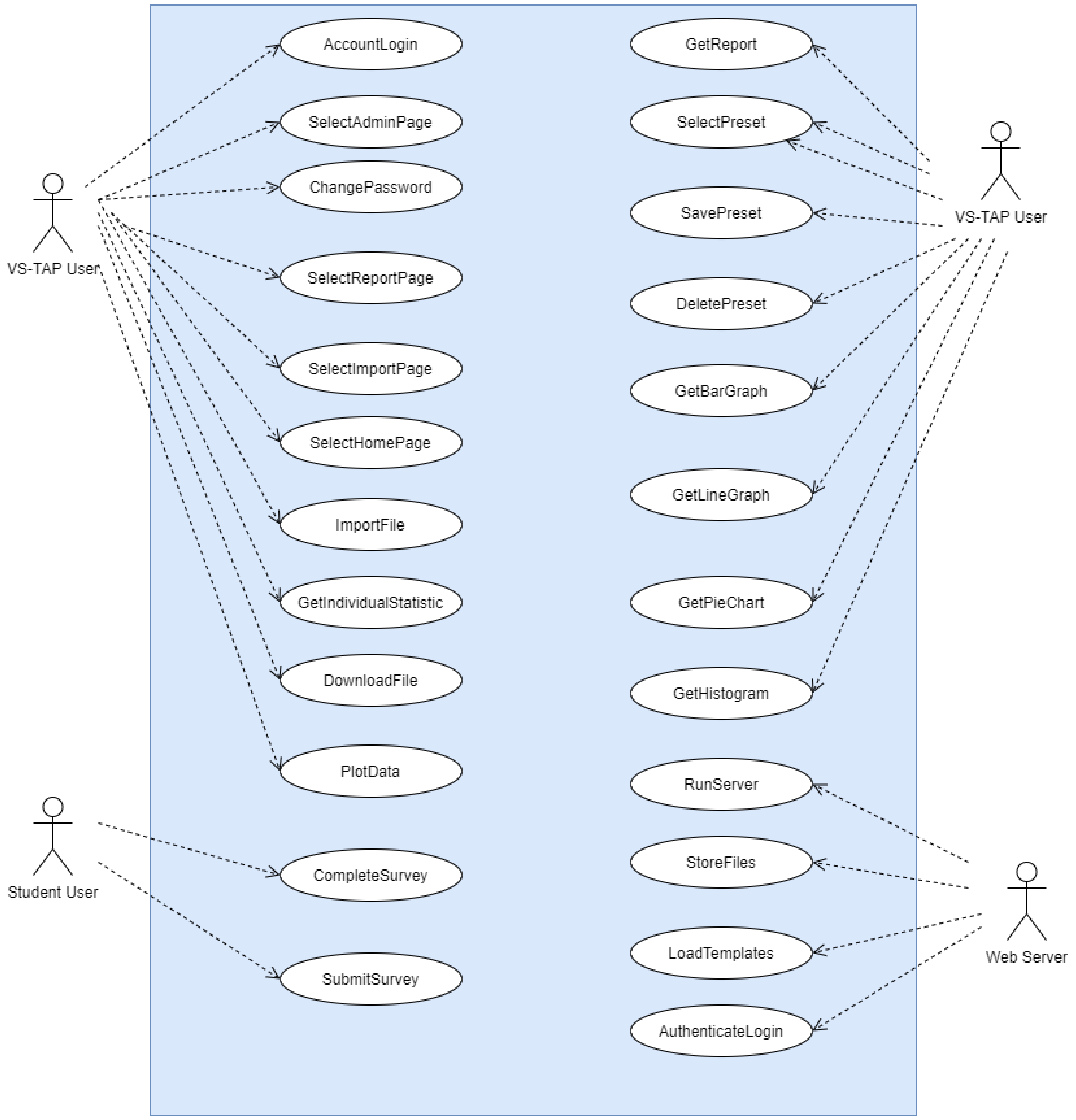


Figure 1: Use case diagram

- **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 Acceptance Criteria and Testing Strategy

3.1 User Stories

User stories are used to verify that the application meets the usability requirements for the end user. The development team worked closely with the employees at the Veteran Services center to come up with a list of user stories that can be broken up into discrete tasks which can be independently tested and implemented. Below is a list of user stories for the VS-TAP application.

- As a user with an existing account I want to be able to log in so that I can create and view reports.
  - When users input valid credentials, they are logged in to the appropriate account.
  - After logging in the user is given access to create reports.
  - User profile settings are stored to their account.
  - Users should be able to input their credentials into text forms.
- As a user without an account I want to be able to create an account so that I can login in the future.
  - Users without accounts can create an account by visiting the create account page.
  - User login credentials are stored in the database, allowing users to login after creating.
  - Users can enter credentials into text forms to create account.
- As a user I want to be able to upload .csv files with visit data so that I can use that data in future reports.
  - Users can visit an upload page and upload a document.
  - If the document is a .csv file with navigate data, it is parsed into a list of visits.
  - Parsed data is inserted into the database so that it can be queried later.
- As a user I want to be able to manually input visit data for visitors without Wolf Cards.
  - Users can visit a manual entry form page.
  - Users can input and submit visit data in a series of text fields.
  - When submitted, the manually entered data is inserted into the database so that it can be queried later.
- As a user I want to be able to create dynamic visualizations to reflect visitor trends.
  - Users can visit the custom reports page.
  - Users are walked through a creation wizard process for data visualization.
  - Users are given the option to save customization fields for reuse.
  - After finalization, a table and graph matching user specifications is generated.
  - Users may download visualized report.
- As an administrative user I want to be able to create accounts for other users.
  - Admins can visit a user creation page.
  - Admins can input account credentials to create an account for other users.
  - The new user credentials are stored into the database.
  - After new account creation, the new user can log in and view the site.
- As an administrative user I want to be able to delete other user accounts.
  - Admins can visit an account list page.
  - From the account list page, users can select individual user accounts.
  - From an individual user account profile, admins can select to delete an account.
  - If an account is deleted, it's entry in the database is removed.
  - After account deletion, that user can no longer log in and view the site.
- As an administrative user I want to be able to edit other user's login credentials and personal information.
  - Admins can visit an account list page.
  - From the account list page, users can select individual user accounts.
  - From an individual user account profile, admins can select to edit an account.

  - If an account is selected to be edited, a set of text forms are presented.
  - If an admin alters the data in the text field from the user's current settings the new information replaces the old field in the database.
  - If a user's login credentials are changed, that user can no longer log in with their old credentials.
- As a user I want to be able to create a bar graph demonstrating the number of visits for each day in a month.
  - When visiting the visualization page, users can select bar graph as an option to generate.
  - After selecting bar graph, users can select usage by date as an option to graph.
  - Users can manually set the colors and scaling for the bar graph.
  - Users can manually chose a range of dates to pull data from.
  - After users select all of the relevant options, they can chose to view the report.
  - Data should be pulled from the data base to generate the report.
  - After selecting to view the report, users are shown a bar graph demonstrating the number of visits for each day in their date range.
- As a user I want to be able to export reports as images and .csv files to include in other files.
  - When generating a report, users should have the option to include a data table in the report.
  - When viewing a report, users should have the option to export each figure as a .png they can download.
  - When viewing a report, if a data table was included, users should have the option to export the table as a .png they can download.
  - When viewing a report, if a data table was included, users should have the option to export the table as a .csv file they can download.
- As a user I want to be able to select a page from a navigation bar so that I can easily change between different pages on the website.
  - A navigation bar with a list of available pages should be visible to users at all times.
  - When clicking on a page from the navigation bar users should be taken to the selected page.
  - When a user clicks on a different page while filling in forms in another page those forms are discarded.
- As a user I want to be able to select log out from the navigation bar so that I can log out and exit the site.
  - Log out should be an option on the navigation bar.
  - When log out is selected the user is taken to the login splash page.
  - If a user is logged out, they have to re-enter their credentials to enter the site.

3.2 Testing Strategy

The benefit of working closely with the project's end users is the efficacy of acceptance and user tests. The VS-TAP team is able to host a project built for the stakeholders so that they can use it and report any bugs or underdeveloped features. These testing strategies are the main strategies employed to test user experience, while automated testing is used to verify that each page of the web-app is accessible. Table 1 outlines some of the testing done.

The Test Type column indicates what category of test that test falls under. The two main categories are automated tests, which are tests that are run programmatically and user and acceptance tests, which involve the end users using the product to make sure it meets specifications. The Target File or Screen column indicates what part of the project is being tested by the test. The Test Data or Situation indicates what environment the project is being tested under. Lastly, the Outcome and Actions Required column indicates what was found and needed to be improved as a result of the testing.

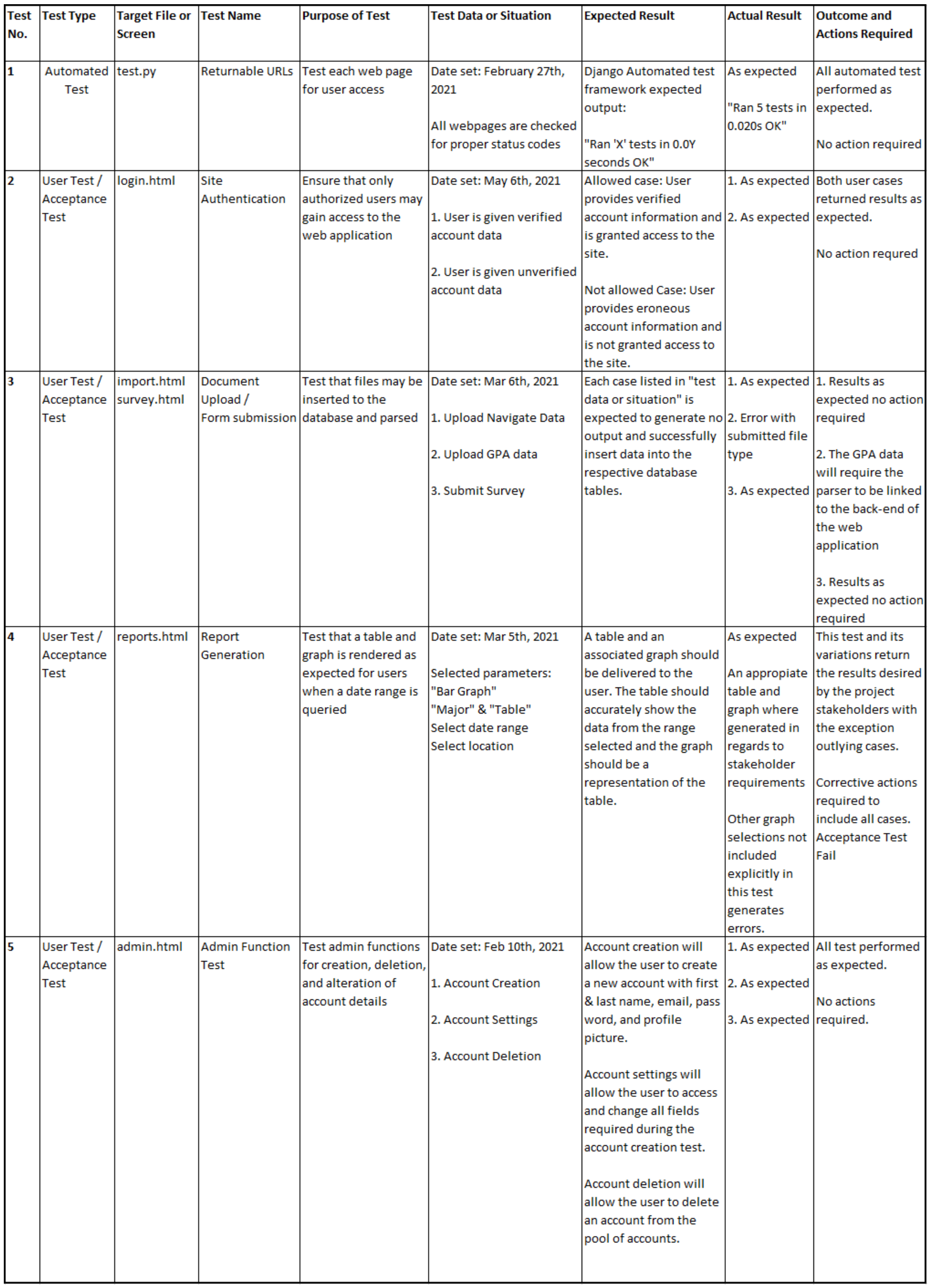
4 Technologies Used

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 [2] uses a concept known as Model-View-Template (MVT), which is based off of the Model-View-Controller architectural pattern. 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

Table 1: Acceptance Test Plan



Test No.	Test Type	Target File or Screen	Test Name	Purpose of Test	Test Data or Situation	Expected Result	Actual Result	Outcome and Actions Required
1	Automated Test	test.py	Returnable URLs	Test each web page for user access	Date set: February 27th, 2021 All webpages are checked for proper status codes	Django Automated test framework expected output: "Ran 'X' tests in 0.0Y seconds OK"	As expected "Ran 5 tests in 0.020s OK"	All automated test performed as expected. No action required
2	User Test / Acceptance Test	login.html	Site Authentication	Ensure that only authorized users may gain access to the web application	Date set: May 6th, 2021 1. User is given verified account data 2. User is given unverified account data	Allowed case: User provides verified account information and is granted access to the site. Not allowed Case: User provides eroneous account information and is not granted access to the site.	1. As expected 2. As expected	Both user cases returned results as expected. No action requred
3	User Test / Acceptance Test	import.html survey.html	Document Upload / Form submission	Test that files may be inserted to the database and parsed	Date set: Mar 6th, 2021 1. Upload Navigate Data 2. Upload GPA data 3. Submit Survey	Each case listed in "test data or situation" is expected to generate no output and successfully insert data into the respective database tables.	1. As expected 2. Error with submitted file type 3. As expected	1. Results as expected no action required 2. The GPA data will require the parser to be linked to the back-end of the web application 3. Results as expected no action required
4	User Test / Acceptance Test	reports.html	Report Generation	Test that a table and graph is rendered as expected for users when a date range is queried	Date set: Mar 5th, 2021 Selected parameters: "Bar Graph" "Major" & "Table" Select date range Select location	A table and an associated graph should be delivered to the user. The table should accurately show the data from the range selected and the graph should be a representation of the table.	As expected An appropiate table and graph where generated in regards to stakeholder requirements Other graph selections not included explicitly in this test generates errors.	This test and its variations return the results desired by the project stakeholders with the exception outlying cases. Corrective actions required to include all cases. Acceptance Test Fail
5	User Test / Acceptance Test	admin.html	Admin Function Test	Test admin functions for creation, deletion, and alteration of account details	Date set: Feb 10th, 2021 1. Account Creation 2. Account Settings 3. Account Deletion	Account creation will allow the user to create a new account with first & last name, email, pass word, and profile picture. Account settings will allow the user to access and change all fields required during the account creation test. Account deletion will allow the user to delete an account from the pool of accounts.	1. As expected 2. As expected 3. As expected	All test performed as expected. No actions required.

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.

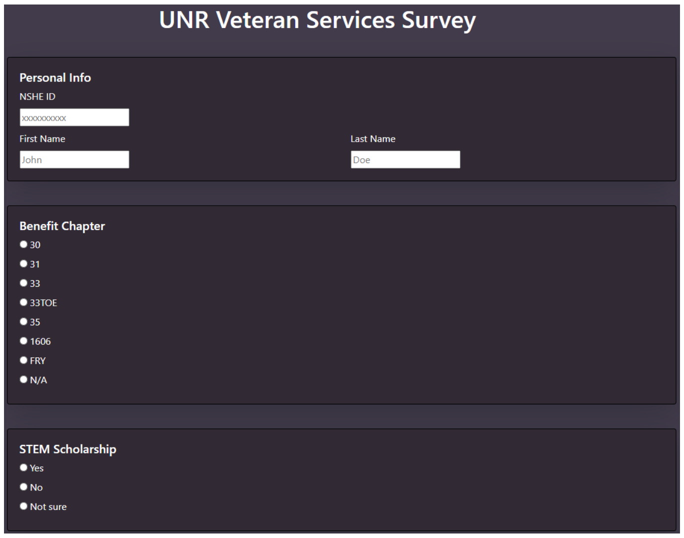
Dash: Dash is a library used for creating detailed and informative interfaces that provide visual reports through Plotly [7]. VS-TAP uses Dash because each visit report needs visual representation, such as a Bar Graph or a Pie Chart. Django contains an extension known as Django-Plotly-Dash. Django-Plotly-Dash 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.

5 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


UNR Veteran Services Survey

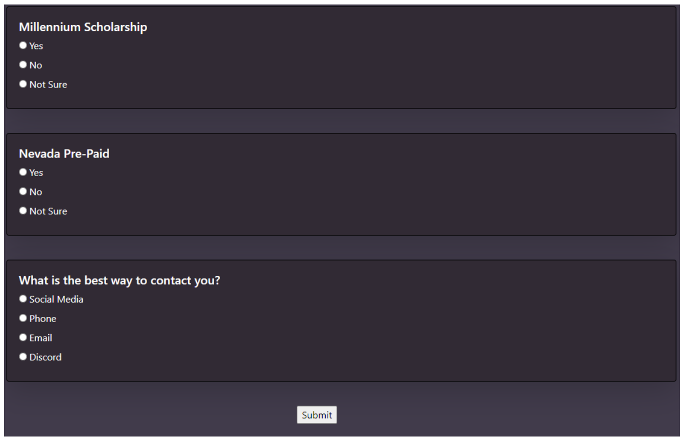
Personal Info
NSHE ID
xxxxxxxxxx
First Name
John
Last Name
Doe

Benefit Chapter
- 30
- 31
- 33
- 33TOE
- 35
- 1606
- FRY
- N/A

STEM Scholarship
- Yes
- No
- Not sure

Figure 2: The UNR Veteran Services Survey requires each visitor's NSHE ID to relate their survey demographics to their visit data

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.

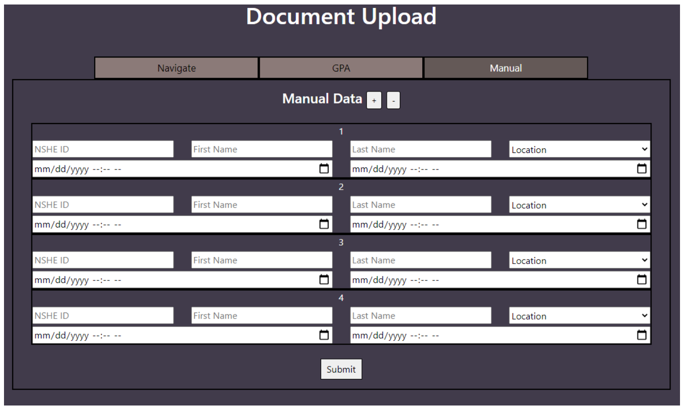

Millennium Scholarship
- Yes
- No
- Not Sure

Nevada Pre-Paid
- Yes
- No
- Not Sure

What is the best way to contact you?
- Social Media
- Phone
- Email
- Discord

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

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


Document Upload

Manual Data + -

1
NSHE ID | First Name | Last Name | Location
mm/dd/yyyy --:-- -- | mm/dd/yyyy --:-- --

2
NSHE ID | First Name | Last Name | Location
mm/dd/yyyy --:-- -- | mm/dd/yyyy --:-- --

3
NSHE ID | First Name | Last Name | Location
mm/dd/yyyy --:-- -- | mm/dd/yyyy --:-- --

4
NSHE ID | First Name | Last Name | Location
mm/dd/yyyy --:-- -- | mm/dd/yyyy --:-- --

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

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 visited the center between the dates of 04/05/2021-08/31/2021.

6 Conclusion and Future Work

The staff of Veteran Services (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 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 achieved through libraries such as Dash Enterprise [8].

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.

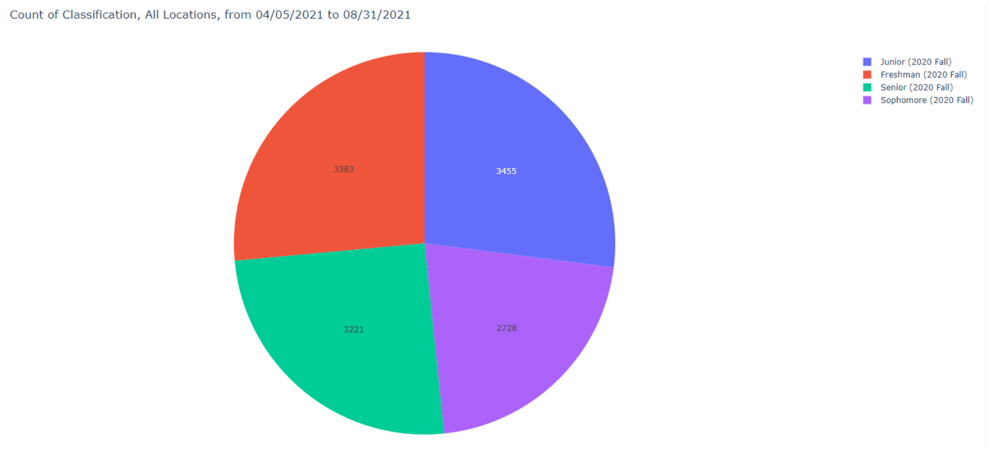
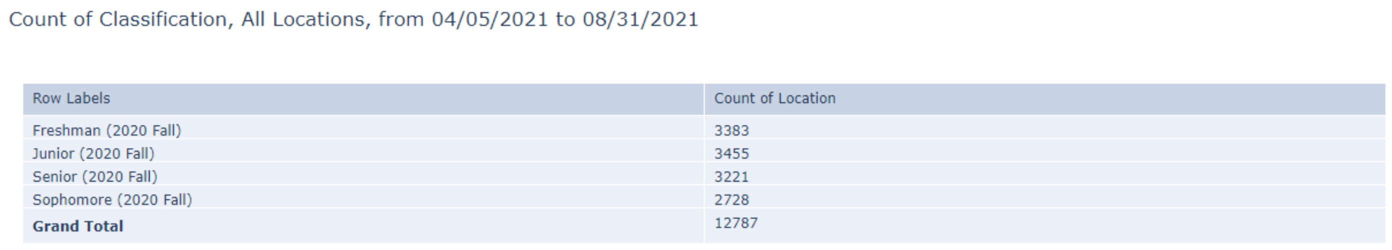


Figure 5: Pie chart representing the amount of students who visit the center and their associated classification year


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 6: Table representing the amount of students who visit the center and their associated classification year

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, Inc. “Employee Time Tracking”. https://www.adp.com/what-we-offer/time-and-attendance/employee-time-tracking.aspx (Last Accessed: 2/3/2002).

[2] Django Software Foundation. “Django Documentation, Release 3.2.4.dev”. https://docs.djangoproject.com/en/3.2/#django-documentation (Last Accessed: 2/3/2002).

[3] Kronos Incorporated. “Kronos Workforce Ready Data Visualizations”. https://www.kronos.com/resource/download/29126 (Last Accessed: 3/7/2002).

[4] Kronos Incorporated. “Time & Attendance System; Time Tracking Software — Kronos”. https://www.kronos.com/products/time-and-attendance (Last Accessed: 3/7/2002).

[5] Linux.org. “Systemd-nspawn - Spawn a Namespace Container for Debugging, Testing and Building at Linux.org”. https://www.linux.org/docs/man1/systemd-nspawn.html (Last Accessed: 2/3/2002).

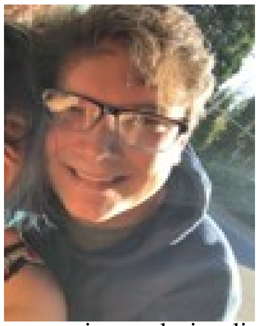
[6] Microsoft Corporation. “Microsoft Excel”. https://www.microsoft.com/en-us/microsoft-365/excel (Last Accessed: 2/3/2002).

[7] Plotly. “Dash Documentation & User Guide — Plotly”. https://dash.plotly.com/ (Last Accessed: 2/3/2002).

[8] Plotly. “Dash Enterprise”. https://plotly.com/dash/ (Last Accessed: 2/3/2002).

[9] Ian Sommerville. *Software Engineering*. Pearson, 10th edition, 2016. https://www.pearson.com/us/higher-education/program/Sommerville-Software-Engineering-10th-Edition/PGM35255.html (Last Accessed: 2/3/2002).

[10] Tableau Software, LLC. “Salesforce + Tableau”. https://www.tableau.com/solutions/salesforce (Last Accessed: 2/3/2002).



Jonathon Hewitt received his BS in Computer Science and Engineering from the University of Nevada, Reno in 2020. His research interests are in Software Engineering, Computer Graphics, Image Processing, and Security. Since he graduated he has pursued a career in Software Engineering and is currently working for a company that creates image processing and visualization tools for the medical and airport security industries.

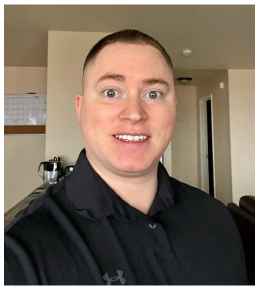
Daniel Hall is an alumni from the University of Nevada, Reno. He received his Bachelor of Science in Computer Science and Engineering in 2021. He also had a minor in Mathematics. His research interest are in Software Engineering, Game Design, Distributed Computing, and Big Data Systems. Since he graduated he has pursued a career in Software Engineering and is currently working for a defense contractor supporting the U.S. Dept. of Defense.

Christopher Parks is an alumni from the University of Nevada, Reno. He received his Bachelor of Science in Computer Science and Engineering in 2019. His research interest are in Software Engineering, Computer Graphics, and Cyber Security. Since he graduated he has pursued a career in Software Engineering and is currently working in a startup in the insurance industry.

Payton Knoch is an alumni from the University of Nevada, Reno. He received his Bachelor of Science in Computer Science and Engineering in 2019. His research interest are in the areas of Software Engineering, Video Games, and Cybersecurity. He has implemented projects to mimic an original network attack and defense model in cybersecurity and he has designed a video game on the Android operating system for personal entertainment. Payton has utilized his skills learned at the university to pursue a career as a software engineer in the insurance industry.

Sergiu M. Dascalu is a Professor in the Department of Computer Science and Engineering at the University of Nevada, Reno (UNR), which he joined in July 2002. He received his PhD degree in Computer Science (2001) from Dalhousie University, Canada and a Master's degree in Automatic Control and Computers (1982) from the Polytechnic of Bucharest, Romania. At UNR he is also the Director of the Software Engineering Laboratory (SOELA) and the Co-Director of the Cyberinfrastructure Lab (CIL). Since joining UNR, he has worked on research projects funded by federal agencies (NSF, NASA, DoD-ONR) as well as the industry. He has advised 11 PhD and over 50 Master students. He received several awards, including the 2009 Nevada Center for Entrepreneurship Faculty Advisor Award, the 2011 UNR Outstanding Undergraduate Research Faculty Mentor Award, the 2011 UNR Donald Tibbitts Distinguished Teacher of the Year Award, the 2014 CoEN Faculty Excellence Award, and the 2019 UNR Vada Trimble Outstanding Graduate Mentor Award. He is a Senior Member of the ACM.

Devrin Lee finished her BS in Computer Science in 2005, and her MS in Information Systems in 2008 from the University of Nevada, Reno. She is a Project Management Professional and a certified ScrumMaster. She has done consulting for small businesses in the IT arena, was the manager of Technical Operations for PCLender, and is currently an Operational Program Manager for Microsoft. Her research interests are in software engineering, product design, and project management.



Nikkolas J. Irwin Nikkolas Irwin received his BS in Computer Science and Engineering from the University of Nevada, Reno in 2020. He currently works for the U.S. Department of Energy (DOE), Office of Inspector General (OIG), Office of Technology, Financial, and Analytics (OTFA) as a data scientist. His current interests include leveraging DataOps, MLOps, and more broadly DevOps to enhance data science workflows.

Frederick C. Harris, Jr. received his BS and MS degrees in Mathematics and Educational Administration from Bob Jones University, Greenville, SC, USA in 1986 and 1988 respectively. He then went on and received his MS and Ph.D. degrees in Computer Science from Clemson University, Clemson, SC, USA in 1991 and 1994 respectively.

He is currently a Professor in the Department of Computer Science and Engineering and the Director of the High Performance Computation and Visualization Lab at the University of Nevada, Reno. Since joining UNR, he has worked on research projects funded by federal agencies (NSF, NASA, DARPA, ONR, DoD) as well as industry. He is also the Nevada State EPSCoR Director and the Project Director for Nevada NSF EPSCoR. He has published more than 300 peer-reviewed journal and conference papers along with several book chapters and has edited or co-edited 14 books. He has had 14 PhD students and 81 MS Thesis students finish under his supervision. His research interests are in parallel computation, simulation, computer graphics, and virtual reality. He is also a Senior Member of the ACM, and a Senior Member of the International Society for Computers and their Applications (ISCA).