Project GoNavy!

Project Final Report

Project Owner:
Erik Shepard

Developers:
Gustavo Hammerschmidt
Jonathan Scott
Erik Shepard

CS441 Software Engineering
Section 01
California State University, San Marcos
11 May 2020

Table of Contents

1 Introduction

1.1 Project Overview

The San Diego Naval Reserve Officer Training Corps (NROTC) is one of 63 consortiums in the country which produces the majority of military officers in the United States Navy and Marine Corps. The composition of the unit consists of an active duty permanent staff, a student staff, and a student body. This large and very diverse program manages over 250 personnel in training and military development.

Many different training events occur frequently over every academic term which are requirements for all students and the statistics from each event must be well documented. Student schedules are also documented to ensure minimal conflicts occur for accountability when a student cannot attend an event. A routing process is in place for a student to request in advance for approval to miss an event should the occasion arise.

Currently, all managing staff utilize stand-alone documents and programs such as Microsoft Excel Spreadsheets and Word documents, Google Sheets, Adobe Acrobat Forms, Blackboard, and email to maintain the large amount of data throughout every term. Errors occur frequently when documenting attendance rosters, sending electronic correspondence, and inconsistent formatting due to misinformation, typos, and incorrect personnel data among other reasons. Much of the documentation that occurs contains consistent information but is reproduced many times, increasing the odds for an error to occur.

A custom website application is desired to be able to encapsulate a central point of data for proper documentation. The application must be accessible and user-friendly for all personnel via the internet. It must also be secure to protect sensitive personally identifiable information (PII).

2 System Functions

2.1 Administrator Functional Requirements

(Requirement Identifier, Priority, Requirement)
AD.01 1 The system shall allow the administrator to register a new user.
AD.02 1 The system shall allow the administrator to set or restrict privileges for any user's access to any methods.
AD.03 1 The system shall allow the administrator to have privileges to all methods.
AD.04 1 The system shall implement separate permissive methods that allow the administrator to authorize individual users the ability:

- to update Demographic Data information for Unit Staff user accounts.
- to update Demographic Data information for Student user accounts.
- to update Service Data for Student user accounts.
- to update Command Data for Student user accounts.
- to update Academic Data for Student user accounts.
- to update Training Data for Student user accounts.
- to update Watchbill Scheduling for Student user accounts.
- to update reasons and approval authority for chit submissions.
- to view Demographic Data information for Unit Staff user accounts.
- to view Demographic Data information for Student user accounts.
- to view Service Data for Student user accounts.
- to view Command Data for Student user accounts.
- to view Academic Data for Student user accounts.
- to view Training Data for Student user accounts.
- to view Watchbill Scheduling for Student user accounts.
- to view all chit requests and comments.
- to approve or disapprove chits.
- to continue chits beyond the approval authority.
- to publish announcements.
- to edit announcements.
- to view announcements.
- to manage the shared document storage.
- to set the application into maintenance mode by replacing the login menu with a maintenance page and removing access to all users for the duration of the maintenance period.

2.2 Chit Functional Requirements

(Requirement Identifier, Priority, Requirement)
CHIT.01 1 The system shall allow a user to submit a request for a chit approval.
CHIT.02 1 The system shall allow a user to cancel a previously requested chit.
CHIT.03 1 The system shall allow a user to edit a previously requested chit.
CHIT.04 1 The system shall allow a user to attach documents during the chit request submission process.
CHIT.05 1 The system shall set the approval authority for the submitted chit in accordance to the rules set based on the reason of the chit.
CHIT.06 1 The system shall implement a new comment log for each submitted chit with a timestamp and the user's name for each comment.
CHIT.07 1 The system shall log a comment at the moment the chit was submitted, a decision is made, the chit was resubmitted, the chit is acknowledged, and/or when the chit was canceled.
CHIT.08 1 The system shall not allow the originator to make any changes to the submitted chit unless the originator initiates the edit method.
CHIT.09 1 The system shall allow the user to 'Resubmit' the chit once the edit method has been initiated.
CHIT.10 1 The system shall restart the approval process to the lowest member in the originator's CoC once the resubmit method has been initiated.
CHIT.11 1 The system shall allow users within the originator's CoC to be able to comment on the chit and make a single decision choice of 'Recommend', 'Not Recommend', 'Approve', 'Disapprove', or 'Rework'.
CHIT.12 1 The system shall sequentially allow each user within the originator's CoC to make a comment and make a decision one at a time beginning from the lowest member in the CoC to the highest member.
CHIT.13 1 The system shall initiate the edit method once a user in the CoC chooses the 'Rework' decision.
CHIT.14 1 The system shall allow the option for the CoC to continue to the next higher member along the CoC beginning at the OI level.
CHIT.15 1 The system shall allow the originator and all users within the CoC to be able to view the requested chit with attached documents and comments.
CHIT.16 1 The system shall include a link on the header of the web page which will allow all users access to the chit system.
CHIT.17 1 The system shall include three views for the chit system: 'New Chit Request', 'My Submitted Chits', and 'Command View'.
CHIT.18 1 In the 'New Chit Request', the system shall allow any user to submit a new chit.
CHIT.19 1 In the 'My Submitted Chits' view, the system shall allow any user to view all submitted chits and the current status for only that user's submitted chits. The system shall also allow the user to edit or cancel any active chits.
CHIT.20 1 The system shall only allow authorized users access to the 'Command View' option.

CHIT.21 1 In the 'Command View', the system shall have a subview for 'My Action' and 'My Command'.
CHIT.22 1 In the 'My Action' view, the system shall display all chits awaiting action for that user, allow the user to view the chit, and allow the user to comment and take action.
CHIT.23 2 In the 'My Command', the system shall list all active chits for all members under that user's CoC.

2.3 Communication Functional Requirements

(Requirement Identifier, Priority, Requirement)
CR.01 1 The system shall allow any user to update their phone number, physical address, and email address.
CR.02 2 The system shall be able to notify users of any published announcements by email to all user email addresses.
CR.03 2 The system shall be able to notify users of any awaiting actions or informational copies of requested chit approvals.
CR.04 2 The system shall allow users to set notification settings to receive emails in real-time, daily digest, or none at all.
CR.05 3 The system shall allow any users to message one or more users in a group chat by selecting recipients individually or by filters of rank, assignment codes, billets, user types, graduation date, or event roster.

2.4 Document Storage Functional Requirements

(Requirement Identifier, Priority, Requirement)
DS.01 1 The system shall have a document storage section to be accessible by all users.
DS.02 1 The system shall have a document storage section to be accessible by only staff members.
DS.03 1 The system shall have a private document storage section to be accessible for chits by only authorized users that have chit review authority.

2.5 Scheduling Functional Requirements

(Requirement Identifier, Priority, Requirement)
SCH.CC.01 3 The system shall implement a Daily Student Log which produces a 'Count Card' of all students by selection of a chosen date.
SCH.CC.02 3 The 'Count Card' system shall display a list of all student's rank, name, assignment code, phone number, approved chits for the particular day, and the chit reason.
SCH.CC.03 3 The 'Count Card' system shall be sorted by the assignment code, then by student's last name.
SCH.CC.04 3 The 'Count Card' system shall include breaks between Companies and Platoons and include counts of total personnel for the Battalion, each Company, each Platoon, how many for each have an approved chit, and the on-hand count (total minus approved chits).

SCH.POW.01 1 The system shall implement a 'Plan of the Week' system.
SCH.POW.02 1 The 'Plan of the Week' system shall have a link for its own section in the navigation menu.
SCH.POW.03 1 The 'Plan of the Week' system shall list a weekly calendar-like view for all events that will occur during that week. The events shall be referenced from any event rosters created with the respective date.
SCH.POW.03 1 The 'Plan of the Week' system shall list all of the duties from the Watchbill for that particular week. Every duty shall include the rank, name, and phone number of all individuals who have duty on that week's Watchbill.
SCH.POW.04 2 The system shall implement a weekly navigation using left and right arrows to view any particular week's Plan of the Week.
SCH.WB.01 2 The system shall implement a 'Watchbill' scheduling system.
SCH.WB.02 2 The system shall have an input for the begin and end term dates by an authorized user only.
SCH.WB.03 2 The system shall allow the authorized user to add various duties and have ltering parameters set to which members are allowed to fulfill that duty.
SCH.WB.04 2 The system shall have a best t populated schedule to allow for even distribution of duties based on the duty parameters.

2.6 Security Functional Requirements

(Requirement Identifier, Priority, Requirement)
SR.01 1 The system shall allow any user to securely login using their username and password.
SR.02 1 The system shall allow any user to change their password and security questions after their current password has been confirmed.
SR.03 1 The system shall allow any user to request their password to be reset if they forgot their password by confirming three security questions.
SR.04 3 The system shall allow the administrator to remove the connection for any particular users at any given time.

2.7 Staff User Functional Requirements

(Requirement Identifier, Priority, Requirement)
STF.01 1 The system shall have a staff profile page that contains Demographic and Students Assigned sections.
STF.02 1 The system shall list in the Demographic section the staff member's username, rank, last name, rst name, middle initial, billet title, gender, phone number, and email address.
STF.03 1 The system shall list in the Students Assigned section all of the student names and rank for who the staff member is assigned to as an OI. If the staff member is not an OI, then all students shall be listed.
STF.04 1 The system shall list additional information in the same line for each student which shall include the student's phone number, school, GPA, and fitness score.
STF.05 2 The system shall include a comment log link for each student line.

This will allow the staff member to annotate comments on the student that is only visible to other staff members. Comments shall be time stamped and also log the last name of the user who logs a comment.
STF.06 3 The system shall include a lter tool to sort or filter the list of students based on a choice of any data points that the system stores information for each student (school, major, GPA, fitness scores, muster points, billets, etc.).
STF.07 3 The system shall include an expansion '+/-' feature to the left of each student name. When the '+' is clicked, the student's prole page will expand below. When the '-' is clicked, the student's profile page will collapse and disappear.

2.8 Student User Functional Requirements

(Requirement Identifier, Priority, Requirement)
SU.01 1 The system shall have a student profile page where Demographic, Service, Command, Academic, Training, and Watchbill data is listed and can be viewed by the individual student user.
SU.02 1 The system shall list in the Demographic section the student's username, rank, last name, rst name, middle initial, assignment identifier, EDIPI, date of birth, gender, program, status, service option, and service contract.
SU.03 1 The system shall list in the Demographic section the student's contact information containing their phone number, physical address, and muster point. The student shall have the option to edit the contact information.
SU.04 1 The system shall list in the Academic section the student's school, major, expected graduation term, and cumulative GPA.
SU.05 1 The system shall list in the Academic section all of the student's current courses with the course ID, title, number of units, weekdays, and times.
SU.06 1 The system shall list the number of counselings, attended club events, and volunteer hours for the current term.
SU.07 1 The system shall allow for the student to maintain a current copy of their transcript in the Academic section. Each upload shall overwrite the previous le.
SU.08 1 The system shall allow for the student to maintain a current copy of their degree plan in the Academic section. Each upload shall overwrite the previous le.
SU.09 1 The system shall list in the Training section the number of club events attended, volunteer hours served, for the current term only.
SU.10 1 The system shall list in the Training section the total number of counselings received.
SU.11 1 The system shall list in the Training section all of the physical fitness test data for the student sorted by date. The most current shall be on top. Data shall include the date, event title, and total score.
SU.12 1 The system shall list in the Training section all of the weigh-in data for the student sorted by date. The most current shall be on top. Data shall include the date, height, weight, body fat percentage, and pass/fail.

SU.13 1 The system shall list in the Training section all of the service training event data for the student sorted by date. The most current shall be on top. Data shall include the date and event title.
SU.14 1 The system shall list in the Training section all of the club event data for the student sorted by date. The most current shall be on top. Data shall include the date and event title.
SU.15 1 The system shall list in the Training section all of the volunteer data for the student sorted by date. The most current shall be on top. Data shall include the date, event title, and the event duration (hours).
SU.16 1 The system shall list in the Training section all of the counseling data for the student sorted by date. The most current shall be on top. Data shall include the date, reason counseled, and name of person counseled by (rank and last name).
SU.17 1 The system shall list in the Training section all of the mentoring data for the student sorted by date. The most current shall be on top. Data shall include the date and name of the mentor (rank and last name).
SU.18 1 The system shall list in the Command section the student's CoC from highest to lowest. The data should include the command level number, billet title, rank and last name, and email address.
SU.19 1 The system shall list in the Command section all of the billets held by the student sorted by term. The most current shall be on top.Data shall include the term and billet title.
SU.20 1 The system shall list in the Completed Chits section all completed chits data for the student sorted by the chit requested begin date.
The most current shall be on top. Data shall include the date, chit reason, and status.
SU.21 3 The system shall have the student's profile photo in the Demographics section. The photo shall have an option to upload and replace the photo.

2.9 Training System Functional Requirements

(Requirement Identifier, Priority, Requirement)
TS.01 1 The system shall have a Training section to be accessible by all users.
TS.02 1 The system shall allow any authorized users to initiate a new training event.
TS.03 1 For every new training event, the system shall allow the user to select an event type, which will consist of 'Club Event', 'Community Service Event', 'Physical Fitness Event', 'Weigh-In Event', or 'Other Training Event'.
TS.04 1 The system shall allow options to choose from for the event title once an event type has been selected.
TS.05 1 The system shall allow the user to enter the following items for each new event: 'event date', 'event owner', 'event begin time', 'event end time', and an 'attendance roster'.
TS.06 1 The system shall calculate the duration of the event from the begin and end times and display to the user in units of minutes.
TS.07 1 The system shall allow the event owner to edit the event at any time.

TS.08 1 The system shall allow the attendance roster to be populated with multiple names. The attendance roster shall have an option to be open to all users to sign-up or only for the event owner to populate.
TS.09 1 The system shall allow the event owner to add additional columns to the attendance roster with custom column headers.
TS.10 1 The system shall automatically create additional columns on the attendance roster based on the event type and title.
TS.AL.01 1 The system shall maintain an award log in the student's training section. The log shall include the date of the award and the type of award.
TS.CL.01 1 The system shall be able to document a counseling log history that includes the date that a counseling was held, the name of the counselor, the name of the member being counseled, and the reason for being counseled.
TS.CL.02 1 The system shall display all counseling logs in the student's training section for counselings if the user's name is the member being counseled.
TS.CL.03 1 The system shall maintain a count of the total number of counselings for every student user. This counter shall be displayed on the student's home page.
TS.ML.01 1 The system shall be able to document a mentorship log history that includes the date that the session was held, the name of the mentor, the name of the member being mentored, and items that were discussed.
TS.ML.02 1 The system shall display all mentorship session logs in the student's training section for mentorship if the user's name is either the mentor or the member being mentored.
TS.PF.01 1 The 'Physical Fitness Event' type shall have sub-events [exercises] based on the selection of the event title.
TS.PF.02 1 The system shall allow the administrator to modify the sub-event titles for each event title under the 'Physical Fitness Event'.
TS.PF.03 1 The system shall allow each sub-event to record a score that is an integer.
TS.PF.04 1 The system shall be able to calculate the total score for each physical fitness event, which will be a sum of the sub-event scores.
TS.PF.05 1 The system shall automatically add columns to the attendance roster for the sub-events and scores.
TS.WI.01 1 The 'Weigh-In Event' type shall allow the system to record member's height, weight, body fat percentage, and pass/fail. The rst three items will consist of double precision data types.
TS.WI.02 1 The system shall automatically add the member's age and gender to the 'Weigh-In Event'.
TS.WI.03 1 The system shall automatically add columns to the attendance roster for age, gender, height, weight, body fat percentage, and pass/fail.
TS.WI.04 1 The system shall allow the event owner of a 'Weigh-In Event' to record the data for each member at any time.
TS.WI.05 3 The system shall allow the event owner of a 'Weight-In Event' to add 'Body Tape' data that includes double precision data types of 'Neck-1', 'Waist-1', 'Hip-1','Neck-2', 'Waist-2', and 'Hip-2'.

TS.WI.06 3 The system shall allow the 'Body Tape' data to automatically calculate the 'CV' by averaging the two similar values of each from TS.WI.05, adding the hip and waist, then subtracting the next value.

2.10 Implemented Functions

2.10.1 Completed functions

- **get_term($requested_date)** returns the requested termID (i.e. the semester)
- **get_user_cmd_billetID($userID, $term)** Returns the billetID (primary key of billets Table) of a given user ($userID) for a given term($term) for only command billets.
- **get_all_user_billets($userID, $term)** Returns an array of billetIDs (primary key of billets Table) for a given user ($userID) for a given term($term) for all billets.
- **get_next_Coc($origin_userID,$userID,$term)** Returns the userID (primary key of users Table) for the next higher billet in the given user's ($userID)
- **get_aid($userID,$term)** Returns a string of the Assignment Identifier Code of the company/platoon/squad for the user ($userID) of a given term ($term).
- **get_CoC_up($userID,$term)** Returns a table of the user's ($userID) Chain of Command.
- **get_OI($userID)** Returns the userID (primary key of users Table) of the given user's ($userID) assigned OI.
- **get_user_cmd_billet($userID, $term)** Returns a string of the billet title of the given user ($userID) for a given term ($term).
- **get_user_collateral_billet($userID, $term)** Returns a string of the collateral billet(s) held by the user ($userID) for a given term ($term).
- **get_user_rank_name($userID)** Returns a string of the ($userID)'s abbreviated rank last name, first and middle initials (ex: LDCR Last, F.M.)
- **get_user_dob($userID)** Returns a date value of the ($userID)'s date of birth.
- **get_user_gender($userID)** Returns a string of the ($userID)'s gender.
- **get_personal_info($userID)** Returns a table displaying the user's personal information.
- **get_current_courses($userID)** Returns a table displaying the ($userID)'s current courses for the current term only.
- **new_academic_term($userID,$term,$grad_term,$major)** Creates a new academic term record for the ($userID) and returns the academicID.
- **insert_course($userID,$term,$cid,$title,$units,$days,$timebegin,$timeend,$campus,$course_num)** Creates a new academic term record for the ($userID).
- **find_academic_term($userID,$term)** Retrieves the academic term record for ($userID) and returns the academicID.
- **update_academic_term($academicID, $item_to_update, $value)** Retrieves the academic term record for ($userID) and returns the academicID.

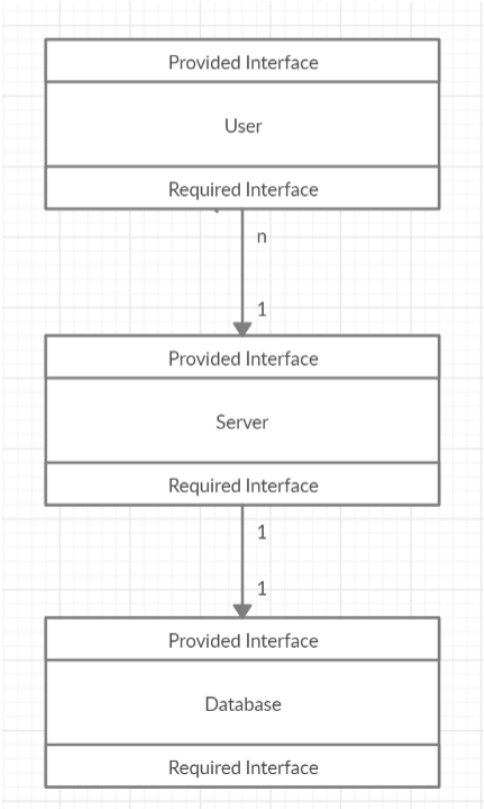
- **set_OI($userID, $OI)** Updates the assigned OI ($OI) for a given user ($userID) with the OI's userID.
- **insert_new_term($semester,$begin,$end)** Creates a new term record.
- **assign_billet($term, $billetID,$company,$platoon,$squad,$userID)** Assigns a billet to a user for the given term.
- **is_cmd_check($billetID)** Returns true or false to check if a billet is a command type billet.
- **extended_personal_info($userID)** This function returns personal-information-div additional information in a shape that I could easily extract with the functions I already had on my PHP related-files Homepage folder.
- **extended_command_info($userID)** This function returns command-information-div additional information in a favorable-to-extraction shape.
- **extended_academic_info($userID)** This function returns academic-information-div additional information in a favorable-to-extraction shape.

2.10.2 Incomplete functions

- **get_pending_chits_by_user($userID)** Returns a table of all chits of pending status by a user.
- **add_student($user_array)** Adds a new student type user record to the database.
- **add_staff()** Adds a new staff member type user record to the database.
- **update_phone($userID, $new_phone)** Updates the phone number of an existing user.
- **update_email($userID, $new_email)** Updates the email address of an existing user.
- **update_address($userID,$new_street,$new_apt,$new_city,$new_residence)** Updates the physical address of an existing user.
- **update_muster($userID,$new_muster)** Updates the muster point of an existing user.
- **update_password($userID,$new_password)** Updates the password of an existing user.
- **update_photo($userID,$new_photo)** Updates the photo of an existing user.
- **update_event()** Updates the event information of an existing event.
- **get_event_roster($eventID)** Retrieves the attendance roster of an existing event.

3 Architecture

This section includes the high-level architecture of the system. An architectural diagram is required. It should be followed by descriptions of the main components of the system. Interface definitions are not required since they are already covered in your Assignment 3.



To solve this problem, we have decided to make a software based on the client-server architecture. There is a user: which, in the client-server model, would be the client. The user is responsible for taking care and managing the software, as well as it can give orders to other users and manage its problems through the application, some functions will be available to some users and others will not, this will be defined according to the user rank: that will dictates if the user is allowed or not to make such operation. Users use the application to solve its problems, check its information, etc. The server is the part of the application that will be running on a server-dedicated computer, it will provide interfaces for its users and access the database to manage data. The database is the part of the software that will save all the application data.

4 Implementation

4.1 Implementation Technologies

- OVH Cloud was used as a hosting provider.
- Name.com for the domain name.
- Jenkins was used to deploy every commit to the master branch to the server.
- PHP was run on the server side using MySQL for database storage.
- Client Side was made of HTML/CSS/Javascript using libraries Jquery, Jquery UI, Knockout.js, and FullCalendar

4.2 Implementation Tasks

Every team member was held responsible to implement some html pages and the implementation of the items related to each of our pages were each one our own responsibility. Gustavo was responsible for the homepage and index pages, Erik was responsible for the training and chits pages, Jonathan was responsible for the communication, documents, leadership, and watchbill pages. Jonathan continued to take on more pages as he was able to complete each page more quickly than Gustavo and Erik. As the index page was a priority at the beginning, Gustavo put his effort into making the authentication of the user at a higher importance to allow the team to pass through without having to worry with logging in every time. Gustavo drew the homepage elements on the html file and then marked what information was to be extracted from the database in order to make it functional. Then Gustavo moved all his elements to the PHP files and added a javascript to make the bridge between the user and server modules work. This was so that his PHP files would be called by the JS files, and they would get the information from the database then return the data to the JS files. Then the JS files would be able to append the data into the html files. The team followed this logic for every other html file cited above. Initially we did not have a database set, so we fixed the returned values on the phps as variables and returned generic values to test this bridge communication between those modules. Jonathan started to make the database when the project was on its second development week. Once the tables containing the most requested values were set, we populated the tables with test users and test information. Then, Gustavo was able to make the server-database bridge, and he made common functions to access and retrieve the database information: for some scenarios, they were really useful and for others the team prefered to use context-bounded access functions to access it. When we had all of the three modules initialized, the team work flowed easily, and Gustavo had time to spend on the authentication part - which he finished within the same day that he started. He also improved the front-end design that day and it has been the same to the end of the project. Gustavo made more PHPs to help to implement the connection to the database for the team and finished the homepage using these

methods. Jonathan finished the database and had a few small problems with the languages used and the application state, he finished his html and js files, as well as his PHP files, in the same manner that Gustavo had made the homepage, which makes the application more concise and easy to understand. When Erik had problems while making his files, the team has managed to assist him throughout the development, and he has finished all his files basic functions and adjusted his coding to our project structure. After finishing the server main functions and communication bridges between the modules, Erik used his time to populate the database with the expected information-to-be for the presentation, and we polished the html elements to better append the information. The database was made with MySQL and is running on a phpMyAdmin server, the server establishes a connection to the database every time it wishes to communicate, the server was made using the Xampp application as base to help us with the server logic. To ease the network programming: Xampp creates a local server on the machine, which allows making web applications easier. The user interface was inspired by the mock-ups that we made for the first presentation. Then changes were added to them to make the pages more visually pleasing for the user. We used GitHub to work simultaneously on the project and to be able to modify files while avoiding conflicts. The group used WhatsApp as a communication tool for working on the program and to debate ideas for its development. All members have had a contact with each module of the project, once we have split every module work in a way for every part to be independent of the other, except for the database, in which Jonathan and Erik had spent more time constructing it, while Gustavo made adjustments to it later. The server consists of many php files and they are more event-bounded than object-oriented, which means that the server has many files used to solve a specific context problem; but some files follow a hierarchy, because they use other php file functions to access the database, making the server context-driven and very dependent. This approach was chosen because it fitted with the project javascript logic and the way things interact in our application one the client-side. Unfortunately, the application's front-end suffered a little bit with our lack of time, thus not being finalized the way intended for it to look like; but we decided to prioritize functionality over visual design.

4.3 Consistency

We were unable to implement all of the requirements that we had originally planned due to the scope of the project being much larger than anticipated. The implementation that we were able to complete was consistent with the design for the most part but we had found some inconsistencies during the development. A lot of user interface requirements related to style of the page and user interaction were modified with some new ones added. We found that what we had was not sufficient to satisfy our needs. Previously, we have conceived the application as having two types of clients, the regular user and the administrator; for time reasons and development work, we decided to remove the administrator module from the application architecture and to think of it as having only one user type. We had some difficulties when implementing the download of files functionality via the pop-ups on the homepage; because we

had to finish other functionalities, we have left that out of the final version. Our student module was simply renamed to user.

4.4 System Availability

GitHub: https://github.com/scott141csu/NROTCWeb

Demo Video:
https://github.com/scott141csu/NROTCWeb/blob/master/Final%20Demo.mp4?raw=true

Live Website: https://arcanist.games/
Username: testuser
Password: 123123

MySQL Database: https://phpmyadmin.arcanist.games
Jenkins CI Server: https://jenkins.arcanist.games/
Username: monty
Password: navy84576!

5 Project Management

5.1 Developer Contributions

Gustavo Hammerschmidt	● Github project management ● General use PHP files ● Server general structure ● HTML elements and divs pertaining to many pages ● Homepage HTML, JS, CSS and PHP files ● CSS general page layout stylesheets ● Helped the team with JS and PHP related problems ● Index page authentication and cookies ● Renamed Database columns and splitted tables
Jonathan Scott	● MySQL, Jenkins, and PHP Cloud Server setup ● Document Storage Feature ● Communication Feature ● Watchbill UI ● Calendar Feature ● Leadership Feature
Erik Shepard	● Project Owner ● Database Relations ● Database Table attributes ● Chit Feature ● Training Feature ● Function designs

5.2 Developer Challenges

Gustavo Hammerschmidt	● Project complexity provided difficulties when coding the ideas, given that I did not know much about NROTC team and technical terms. ● I had some experiences with web development and was able to help the team with their problems, but the project size was always increasing and I have found it difficult to generalize solutions to recurrent problems. ● I am not good with user interface design, but I had to make some basic design to explicitly delineate the position and format of divs to be able to start modeling an effective back-end to the app; unfortunately, we haven't had more time to give a better look to the application's front-end side. ● I was excessively working on other classes' projects and assignments; sometimes, I did not have the time on a week to spend working on the project.
Jonathan Scott	● Huge scope, not enough time. ● Conflicts over implementation details and which libraries for the group to use, compromised with Gustavo and used his initial implementation
Erik Shepard	● Had the least amount of programming experience (none in html, php, and javascript). Learning the programming languages was not difficult but being able to use the structure set up by Gustavo and Jonathan was difficult to figure out. I had to try to communicate with them to figure out how to make it work, but their solutions didn't always work for the application I was working on. ● Difficult to communicate between team members due to COVID. Gustavo was on a different time zone and there were often long delays between messages using Whatsapp. ● Realized that the scope of the project was much more intensive than we anticipated. We tried to focus on each person completing the modules one at a time.

6 Conclusion

Overall, the team is satisfied with what we have achieved and the professionalism with which we have handled any conflicting problems among each other. In the beginning, it was a little bit awkward, because we did not know each other and it was a first-time web application for half of the team. We all agree that we have reached somewhere between 85 to 90 percent of what we had originally intended to do.

The server and the database modules are the ones we are most satisfied with, because we had learned a lot of PHP and MySQL functionalities while working on them. These functionalities were what we nearly completed. There was, at the beginning, an intention to make an administrative user that would be able to insert and manage other users, but that would have increased the project's complexity and was cutted of it because of development time reasons.

The user interface was made from our mockups and it has not had much emphasis throughout the development, which lead to us being unsatisfied with the final result, asfor the front-end part. The front-end would have looked much better iif we had the time to spend on it, but as we were all more experienced as back-end programmers, the structure was much delightful for us to see it working properly. The team agrees that there were not many difficulties related to team cooperation and development, actually we could not agree more that this has been one our best experiences when it comes to project effectiveness and division of tasks. Overall, we are very satisfied with the progress that we did make, although we all wanted to see the project through to the end.

7 References

Libraries:

JQuery - https://jquery.com/
JQuery UI - https://jqueryui.com/
Knockout JS - https://knockoutjs.com/
FullCalendar - https://fullcalendar.io/

Cloud Service - http://ovh.ca/

8 Definitions, Acronyms, and Abbreviations

- Billet. A job position that a member has that requires specific tasks that the member must accomplish.
- Chit. A student’s request for various reasons such as to miss a required event, take time off, not wear uniform, etc. The request is sent up the student's CoC for approval.
- CoC - Chain of Command. A hierarchy of leadership, composed of many levels.
- Duty. A position, usually regarding safety, with unique tasks that is a requirement for logistical reasons to accomplish a job.
- MECEP - Marine Enlisted Commissioning Education Program. A Marine Corps education program specifically for outstanding active duty enlisted personnel to complete their undergraduate education for the purpose of earning a commission as a Marine Corps Officer.
- Muster Point. Location point for a physical meeting.
- NROTC - Naval Reserve Officer Training Corps. A military officer training program that consists of both civilian and enlisted active duty members as full-time college students and managed by a permanent active duty staff.
- OI - Officer of Instruction, also known as an Officer Instructor. A permanent staff member that is an officer and supervises students in their development to become an entry level military officer.
- PII - Personally Identifiable Information. Sensitive information that can be collected in parts to identify a person that could be used for fraud by an unauthorized individual.
- Rank Structure. A hierarchical structure to identify persons with the most experience and authority. Rank is identified by either pay grade or by title. Pay grades are identified by O1 through 09 for ocers and E1 through E9 for enlisted personnel. Titles vary throughout different military branches and may be abbreviated such as 1/C, 2/C, 3/C, 4/C, Sgt, SSgt, GySgt, Capt, LT, and CAPT.
- Service Option. Designation for the branch of military service where a student is pursuing to become an officer.
- Service Program. Signifies which program provided funding for the duration of the student's tenure at the NROTC.
- STA-21. A Navy education program specifically for outstanding active duty enlisted personnel to complete their undergraduate education for the purpose of earning a commission as a Navy Officer.
- Watchbill. A schedule which members will be required to fulfill a certain position, usually regarding safety. These positions are called 'duties'.