

JONATHON BOURKE

C00242865

Project Report

## CONTENTS

Abstract .....	2
Description of Submitted Project .....	3
Assets .....	5
Policies .....	6
Vulnerabilities .....	7
Playbooks .....	8
Ticketing System.....	9
Ticket Object .....	9
Queue .....	10
Vulnerability Metrics.....	10
Authentication .....	12
Django Administration .....	12
Web Application .....	12
Conformance to specification .....	13
My initial experience .....	14
Personal Learning .....	14
Project Review .....	14
Acknowledgements .....	16

## ABSTRACT

This document outlines the outcome the finalized vulnerability management tool and how the brief requirements were satisfied. In the process of describing the finalized product, I will describe my successes in development and summarize the challenges faced in achieving the end-result. While discussing the obstacles encountered, I will mention the solutions to referenced points and how they have affected the final submission.

## INSTRUCTIONS

Clone the Project from GitHub: [Source](#)

### Django

Install Virtual Environments: **pip install virtualenv**

Create your Virtual Environment. Name does not matter: **virtualenv "name as you like"**

Install Django: **pip install Django**

Install Project Dependencies: **pip install -r requirement.txt**

Navigate to the project file "settings.py" and update the database settings depending on the technology and credentials you are using. MySQL Example below:

```
DATABASES = {  
    'default': {  
        'ENGINE': 'django.db.backends.mysql',  
        'NAME': 'vuln_database',  
        'USER': 'vuln_user',  
        'PASSWORD': '*****',  
        'HOST': 'localhost',  
        'PORT': "",  
    }  
}
```

Navigate to folder with manage.py in CLI and run commands:

```
$ python manage.py makemigrations
```

```
$ python manage.py migrate
```

Create a superuser to log in to the application: **python manage.py createsuperuser**

Start the server: **python manage.py runserver 8080**

### Angular

Install Node.js: [Source](#)

Run this command: **npm install**

Navigate to Angular project folder in CLI and run the command: **ng serve --port 8081**

Use a browser of your choice to access the application at <http://localhost:8081/login> and login with the Superuser you created

## DESCRIPTION OF SUBMITTED PROJECT

The final product is a platform-agnostic vulnerability management aid that assists in all areas of the vulnerability management cycle. The application supports multiple regular and super users through the application and Django Administration.

The system implements CRUD operations regarding any object that is relevant to the vulnerability management lifecycle. The following objects were supported in the final implementation:

- Assets
- Policies
- Vulnerabilities
- Playbooks

## ASSETS

The screenshot displays a dark-themed web interface for managing assets. The main form is titled "Asset" and contains several input fields and controls:

- ID:** A text input field with the label "ID" above it.
- OS CPE:** A text input field with the label "OS CPE" above it and "lorem ipsum" as placeholder text.
- Risk:** A dropdown menu with the label "Risk" and a downward arrow.
- SLA Hours:** A text input field with the label "SLA Hours" above it and "45" as the value.
- SLA Minutes:** A text input field with the label "SLA Minutes" above it and "23" as the value.
- Hostname:** A text input field with the label "Hostname" above it and "Win1235" as the value.
- Category:** A text input field with the label "Category" above it and "lorem ipsum" as placeholder text.
- CPE URIs:** A list of CPE URIs, each with a trash icon for deletion. The URIs shown are "cpe:2.3:a:google:chrome:\*\*\*\*\*:" and "cpe:2.3:a:intel:proset\_ac\_3168:\*\*\*\*\*".
- Add Vendor:** A text input field with the label "Add Vendor" above it.
- Add Product:** A text input field with the label "Add Product" above it.
- Add Tag:** A button with the label "Add Tag".

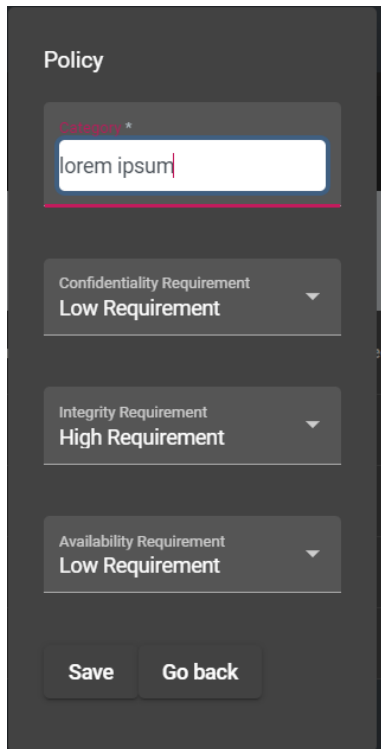
At the bottom of the form, there are two buttons: "Save" and "Close".

Assets combine to form an inventory and were defined under the following key characteristics:

- Hostname
- Operating System defined by CPE system
- Risk

- Expected SLA time
- Hardware and Application list defined with CPE system
- Category

## POLICIES



The screenshot shows a 'Policy' form with the following fields and values:

- Category \***: A text input field containing 'lorem ipsum'.
- Confidentiality Requirement**: A dropdown menu with 'Low Requirement' selected.
- Integrity Requirement**: A dropdown menu with 'High Requirement' selected.
- Availability Requirement**: A dropdown menu with 'Low Requirement' selected.

At the bottom of the form are two buttons: 'Save' and 'Go back'.

Policies can be assigned to assets to assist in defining environmental CVSS values. Policies require the following properties

- Confidentiality Requirement
- Integrity Requirement
- Availability Requirement
- Category

## VULNERABILITIES

**CVE-2022-24086**

CVE  
CVE-2022-24086

SLA

Threat

CWE

Privileges Required  
None

User Interaction  
None

Attack Vector  
Network

Scope  
Unchanged

Attack Complexity  
Low Requirement

Confidentiality Impact  
High Requirement

Integrity Impact  
High Requirement

Availability Impact  
High Requirement

Remediation Level  
Not Defined

CPE URIs

Add Vendor

Add Product

Add Tag

Vulnerabilities are core to the system and are critical to defining both the ticketing system and risk severities. This object contains the most details to encapsulate target vectors, threat characteristics and vulnerable targets.

- CVE Identifier
- SLA
- Applicable CWE
- Privileges Required
- User Interaction Required
- Attack Vector
- Scope
- CIA Impact
- Report Confidence
- Exploit Code Maturity
- Risk
- Description
- Vulnerable CPE Uris

User need not worry about creating vulnerability records manually as there will be automatically retrieved from the NVD database and potentially matched against Assets with affected CPE Uris.



## PLAYBOOKS

Playbook

Category

Patch Validation

Patch Verification

Rollout

Notes

Describes the path vulnerability remediation will take through the system. Offers the default playbook at baseline that follows core patch management methodologies. Adding values modifies the group responsible for that stage of remediation.

- Category
- Patch Validation
- Patch Verification
- Rollout
- Notes

## TICKETING SYSTEM

I will provide a high-level overview of the ticketing system

### TICKET OBJECT

INC#76

Acquirement Validation Verification Rollout

Assets

Jonathons Laptop Jonathons Laptop

Add Asset

Progress: New

Assigned To: unassigned

Group: Patch-Acquirement

Time Started: 2022-02-14T22:18:17.941870Z

CVE: Microsoft Word

Base CVSS: 10.00

Environmental Score

Temporal Score

Ticket Progress

Patch Acquired

Patch Validated

Patch Verified

Patch Rollout

Tickets are the components that house all necessary data regarding the remediation of a vulnerability. Creating the ticket object was a prerequisite for creating remediation flow through the system. The properties of a ticket are too long to list here in the document as they provide:

- Remediation Characteristics
- Threat scores
- Ownership Properties
- Affected assets
- Vulnerability attributes
- Progress features
- Logging

A ticket is created by selected a known vulnerability in the system. Using the CPE matching system, assets are automatically added. Threat scores are calculated on creation through relational data. As the ticket flows through the system both progress and timestamps are logged. Results of the successes of the ticket are logged at closure.

## QUEUE

The queue allows users to view all tickets in their system lifecycle at various patch management stages. Interacting with a ticket will provide further information to read and edit.

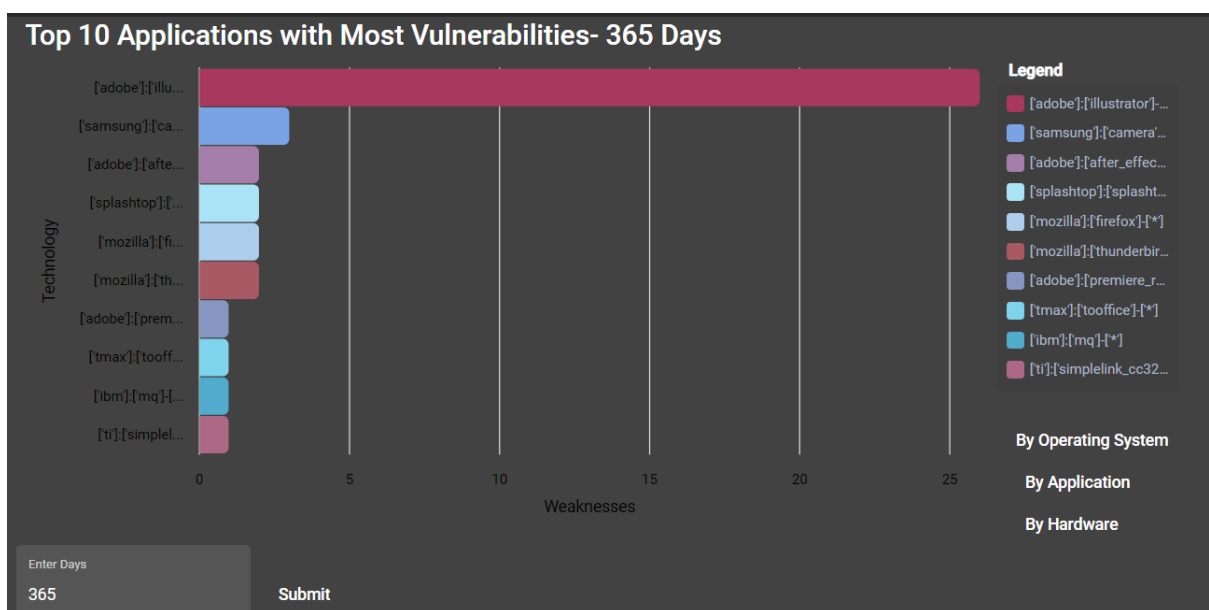
All Tickets							
Search							
All Tickets	Patch Acquisition	Patch Validation	Patch Verification	Patch Rollout	My Tickets	Closed Tickets	New Ticket
ID	Time Started	SLA	Progress	CVE	CVSS		
67	2022-02-14T20:51:06.649488Z	2022-02-14T21:51:06Z	New	CVE -1234	6.30		
68	2022-02-14T21:04:22.067187Z	2022-02-14T23:04:22Z	New	CVE -1234	6.30		
69	2022-02-14T21:07:20.915929Z	2022-02-14T22:07:20Z	New	CVE -1234	6.30		
70	2022-02-14T21:13:21.599144Z	2022-02-15T03:13:21Z	New	CVE -1234	6.30		
74	2022-02-14T22:03:32.302739Z	2022-02-15T14:17:40Z	In Progress	CVE -123456	5.90		

Items per page: 5 51 - 55 of 68 << < > >>

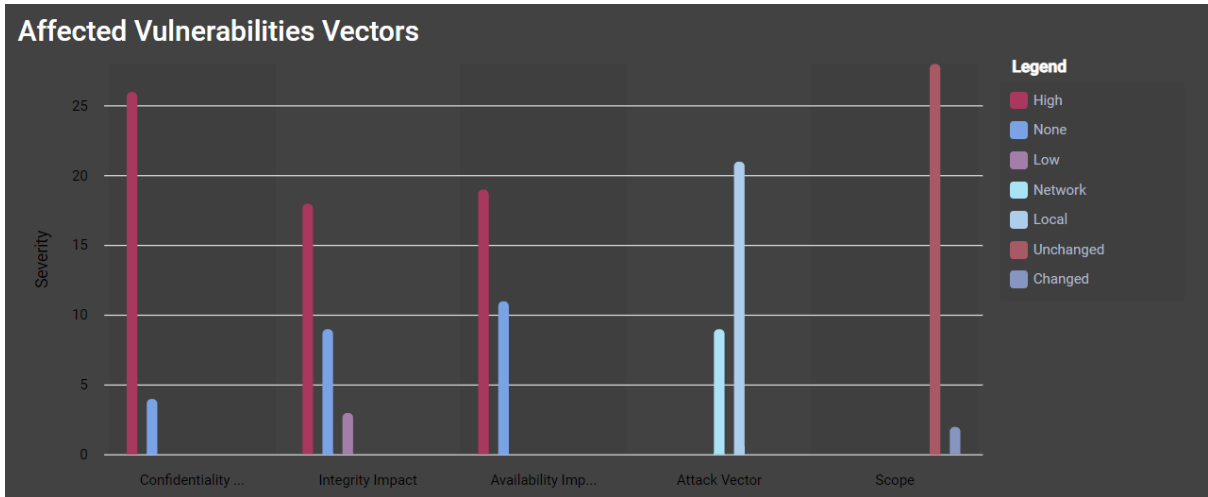
## VULNERABILITY METRICS

I was able to successfully provide metrics by creating APIs to retrieve the required data from the backend through ngxcharts. A timeline filter is given to create specific results and control monitoring over the current situation.

A pre-set vulnerability dashboard provides useful visuals like the following.

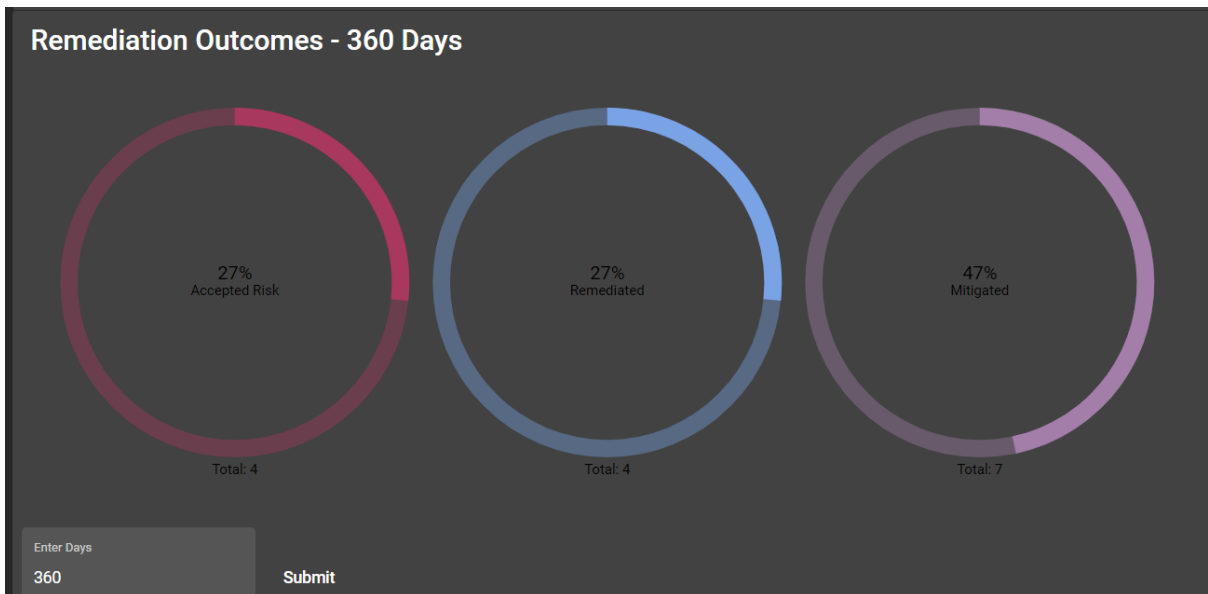


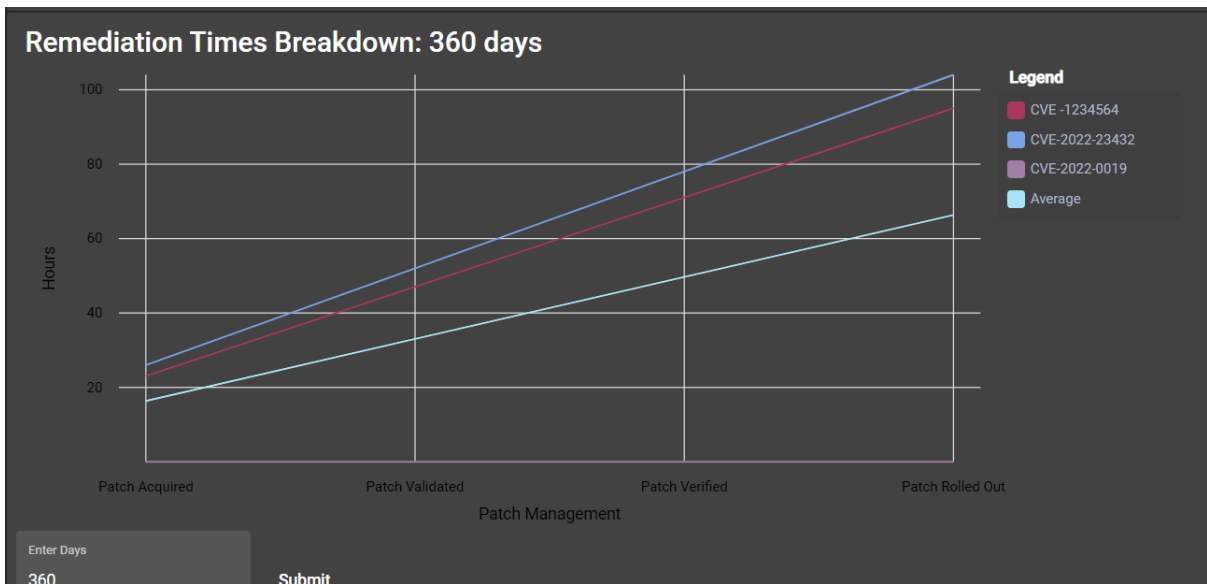
### Affected Vulnerabilities Vectors



A ticket dashboard regarding remediation statistics is also provided.

### Remediation Outcomes - 360 Days





## AUTHENTICATION

The following is a description for implemented authentication mechanisms

### DJANGO ADMINISTRATION

Both Super Users and regular users can be created through the Django Administration script. A Username, email and password is needed.

```
C:\Users\35389\vulnTool\backend\vulnProject>py manage.py createsuperuser
Email: vulnuser3@localhost
Username: vuln3
Password:
```

### WEB APPLICATION

Superusers may use the Web GUI administration to create and manage regular users.

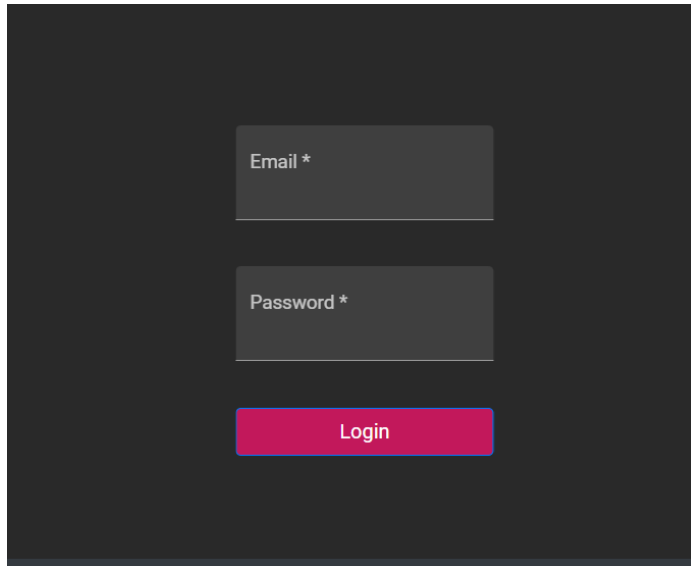
### Administration

Search

Email	Username
vulnuser2@localhost	Jonathon
vulnuser233@localhost	Jonathon33
vulnuser2333@localhost	Jonathon333

Items per page: 5

The application requires authentication to access, and this is provided through both basic and JWT mechanisms. JWT appends a token which can be used to allow a user's session to persist even if they navigate away from the application. This will expire after 60 minutes.

A screenshot of a login form on a dark background. It features two text input fields stacked vertically. The top field is labeled 'Email \*' and the bottom field is labeled 'Password \*'. Below these fields is a bright pink button with the text 'Login' in white.

## CONFORMANCE TO SPECIFICATION

A feature that was not proposed in the initial brief was the CPE URI system to accurately correlate known technologies on an asset and to match affected resources to known vulnerabilities. This later enriched another breakthrough feature that allows the application to parse the NVD vulnerability database, extract the useful information and actively update its definitions through scheduled tasks on the server.

Regarding missing components proposed, the only feature that was completely left out was the integrated vulnerability scanner. This was mainly due to time constraints and the work required to add value to the application as I would have to figure out how I was going to be able to parse the results and make it compatible with my backend. Further, I would need to create additional GUIs to be able to configure, schedule and view the scans. Integration which I initially thought would be a challenge would have been easily achievable with Python Libraries.

Configurable dashboards to not make the final product as similarly it would much more work than I initially thought. The main issue here is Angular graph libraries are very strict when it comes to serialized data they will accept. I was able to overcome this to provide two pre-set dashboards but the extra time it would take to implement in general was not possible in the time allotted.

Due to the proposed features being proposed as optional, I feel it does not take away from the overall achievement of the project

## MY INITIAL EXPERIENCE

When I initially decided on the technologies I would use for this project, I was largely unfamiliar with my technology stack. Regarding Django, I had Python experience but none concerning using the web framework itself. Due to my experience with the language, this was relatively easy to pick up and only required learning Django concepts like Query Sets,

I had no experience with AngularJs or any frontend that utilises typescript. I personally found typescript to have a steep learning curve as many of the reasons that I chose it were very technical to implement. I do believe it was the right choice as the final implementation has a professional feel with the help of Angular features and libraries.

Before starting this project, my skills in web development were limited. Aside from basic PHP knowledge, I had no idea how to configure frontend/backend details or how to bring a hosted application into production and ready to serve clients. This was relatively easy as there is plenty of documentation for Django or Angular to bring it to fruition.

## PERSONAL LEARNING

The most important learning aspect for this project was developing consistency by setting aside time every day to work on the implementation. This allowed me able to finish relatively comfortably and avoid stress relating to trying to meet deadlines. I also learned the importance of allotting time for adequate planning before commencing development in similar projects

## PROJECT REVIEW

My opinion on my project is that it was largely a success as it fulfils the specified requirements in the brief and goes beyond with additional features to assist the vulnerability management lifecycle. Examples are actively pulling vulnerabilities from NVD and the CPE URI system.

The only limiting factor in my view was time. I had high level ideas of how to implement the missing features. To fully implement my original specification, I would likely need another 3-4 months as the integrated scanner required a suitable GUI and making results compatible with my backend.

If I was to start this project again, I would focused more on pulling resources from the internet like what was done in the final implementation with active vulnerability definition updates. I feel there could be more value to extract from the data, but the feature was introduced near the end. I also had to apply my finishing touches and create documentation. This also applies to the CPE system. There is potential to divide this into

application, hardware, and operating system. The Uri fields could be used to generate better matches regarding vulnerable technology. The current implementation matches based on vendor and product, but it would be possible to code version detection such as if it less than or equal to the corresponding Uri value.

My advice for someone who wants to develop their own version of the application would be to spend time learning the technology stack you want to use before developing. I initially tried to immediately code using material I had found online but was unable to adapt the operations into my project. After sitting down for a few days and learning how to code core web application functionality, I was in a better place to continue with the more specific features.

Regarding my technology choices, I believe there isn't a better possible backend choice than Django due to extensive library support for relevant features not available in other languages. Python is an easy language to pick up due to pseudocode-like syntax and query sets which simplify SQL operations.

AngularJs allowed me to achieve a professional application with less work due to its modules. DOM based operations make using the application feel fluid and allow SQL operations to happen through typescript. Other front-end frameworks like React and VueJs may be interesting to investigate however I feel Angular did not have any shortfalls when it came to library support and features.

MariaDB was a great choice personally for the project as it provided me an open-source solution that was very similar to SQL. PostgreSQL may have been the better choice here with Django as many of the model options require it such as Array fields. This would have been very useful for the CPE Uri system as it allows the use of arrays in the database tables. I had to resort to storing data in a CSV format to circumvent this. Something like MongoDB may have been useful as I had to work with serialized data throughout the project. MongoDB allows supports storing data in a Json format.



## ACKNOWLEDGEMENTS

I would firstly like to give a special thanks to my project supervisor Paul Barry for his continuous guidance from the start of my project. Paul's advice has been invaluable throughout the year and kept me on the right track.

I would like to thank the lecturers who I had the pleasure of earning my degree under. If it was not for the accumulated knowledge gathered throughout my bachelor's degree, my finished project would have not been of the same quality.

Finally, I would like to thank my classmates who have made the four years a unique experience and have helped keep me motivated throughout my college journey.