Capture the Flag

Project Solution Approach

**F5 Networks**



**Red Vs. Blue**



Colby Cocking, Logan Gorence, Dung “Richard” Le Hoang, Ethan Isakson, Zhaohu Huang

3/29/21

**TABLE OF CONTENTS**

[I. Project Summary 4](#_Toc68031140)

[II. System Overview 4](#_Toc68031141)

[III. System Architecture 4](#_Toc68031142)

[III.1 Overview 4](#_Toc68031143)

[III.1. Subsystems 5](#_Toc68031144)

[III.2.1. User Interface 5](#_Toc68031145)

[Description 5](#_Toc68031146)

[Design Considerations 5](#_Toc68031147)

[Interface Description 5](#_Toc68031148)

[III.2.2. Frontend API 5](#_Toc68031149)

[Description 5](#_Toc68031150)

[Design Considerations 5](#_Toc68031151)

[Interface Description 5](#_Toc68031152)

[III.2.3. Operator 6](#_Toc68031153)

[Description 6](#_Toc68031154)

[Design Considerations 6](#_Toc68031155)

[Interface Description 6](#_Toc68031156)

[III.2.4. Termproxy 6](#_Toc68031157)

[Description 6](#_Toc68031158)

[Design Considerations 7](#_Toc68031159)

[Interface Description 7](#_Toc68031160)

[III.2.5. User Environments 7](#_Toc68031161)

[Description 7](#_Toc68031162)

[Design Considerations 7](#_Toc68031163)

[Interface Description 7](#_Toc68031164)

[III.2.6. PostgreSQL 8](#_Toc68031165)

[Description 8](#_Toc68031166)

[Design Considerations 8](#_Toc68031167)

[Interface Description 8](#_Toc68031168)

[III.2.7. Kubernetes 8](#_Toc68031169)

[Description 8](#_Toc68031170)

[Design Considerations 8](#_Toc68031171)

[Interface Description 8](#_Toc68031172)

[III.2.8. Auth0 8](#_Toc68031173)

[Description 8](#_Toc68031174)

[Design Considerations 9](#_Toc68031175)

[Interface Description 9](#_Toc68031176)

[IV. Data Design 9](#_Toc68031177)

[IV.1 PostgreSQL 9](#_Toc68031178)

[IV.1.1 User 9](#_Toc68031179)

[IV.1.2 Challenge Set 9](#_Toc68031180)

[IV.1.3 Challenge 9](#_Toc68031181)

[IV.1.4 User Challenge Progress 9](#_Toc68031182)

[IV.2 Operator 10](#_Toc68031183)

[IV.2.1 Environments 10](#_Toc68031184)

[IV.2.2 Challenge 10](#_Toc68031185)

[V. User Interface Design 10](#_Toc68031186)

[VI. State of Project 12](#_Toc68031187)

[VI.1. Tasks 12](#_Toc68031188)

[VI.1.1. Completed Tasks 12](#_Toc68031189)

[VI.2. User Stories 12](#_Toc68031190)

[VI.2.1. Completed User Stories 12](#_Toc68031191)

[VI.3. Epics 12](#_Toc68031192)

[VI.3.1. Completed Epics 12](#_Toc68031193)

[VI. Glossary 13](#_Toc68031194)

[VII. References 15](#_Toc68031195)

[Appendix 16](#_Toc68031196)

[System Architecture 16](#_Toc68031197)

[Kubernetes Component Diagram 16](#_Toc68031198)

[Challenge Sequence Diagram 16](#_Toc68031199)

[Architectural Component Diagram 17](#_Toc68031200)

[State of Project 17](#_Toc68031201)

[Epics 17](#_Toc68031202)

[Epic 1 - User Interface 17](#_Toc68031203)

[Epic 2 - Administrator 18](#_Toc68031204)

[Epic 3 – Challenges 19](#_Toc68031205)

[Schedule 20](#_Toc68031206)

[Project 20](#_Toc68031207)

[Gantt Chart 21](#_Toc68031208)

# Project Summary

We aim to create an application that can be run as a personal or event-based capture the flag (CTF) challenge system. A CTF challenge is a scenario in which a user is given a hacking related puzzle. Solving the puzzle awards the user with points which are then added to their running total. CTF events are held all over the world for hobbyists, or to find the best and brightest hackers to work as cyber security specialists.

The problem with these kinds of CTF events is that they are extremely difficult for beginners and there are not a lot of great places to get started learning how to hack ethically. Our intent with our CTF platform is slightly different than finding the best hackers so we can hire them at our company. Instead, we aim to lower the barrier of entry into ethical hacking. The content we are designing is targeted at a much younger audience than most CTF applications and provides the user with the tools needed to learn how to perform the actions required to overcome the challenges designed by our team or other third parties.

# System Overview

The stakeholders in this project are the Red Vs. Blue team from WSU who are impacted by this project in the sense that we are receiving a grade for it, F5 industries as the sponsor and financial backer of this endeavor, and any potential users who would like to further their knowledge of cyber security. Specifically, inexperienced users are a larger stakeholder than experienced users at this point as our system is targeted at a more inexperienced audience.

Each user will have an account they are associated with. Once logged in, they will have access to the available challenges. Each challenge will be spun up in its own group of Kubernetes pods where the user will have access to their own environment, safe from outside influence. The environment will contain any learning materials necessary for a particular hacking challenge and if any targets need to be spun-up for a challenge, they can be created alongside the same pod and given the same network policies to allow interactions between them. Kubernetes can be run on a public cloud or bare metal for self-hosted installations.

There will also be support for adding challenges and challenge sets to the platform in the future. To help with this, we are creating documentation and schema that contributors will need to follow if they want their challenges to be able to integrate into our platform.

# System Architecture

## III.1 Overview

For our system architecture, we are utilizing a microservice-based architecture. Microservices are a very popular means of creating cohesive and decoupled components that work well in a cloud-native environment. One of the goals that was set out by our sponsor for this project was to make the player’s barrier-to-entry as low as possible, and one of the core ideas here is to allow the player to have all the tools they need in an environment that is hosted by the platform. Here, we chose Kubernetes to provide what we need for hosting environments, as well as hosting our platform’s services too.

Microservice-based architectures instinctively have low coupling and high cohesion patterns. Each microservice performs a particular task, like a module or package in a traditional monolithic application. One of the problems to solve in a microservices-based architecture is how your services talk to one another, both in terms of what protocol they speak and how they identify and discover one another. First off, we are using gRPC to facilitate the connection between microservices. gRPC­­­­­­ is an open-source RPC framework written at Google, based on their internal library, Stubby. To solve the problem of service discovery, we are utilizing Kubernetes. Kubernetes provides multiple methods of service discovery, both environment variables and DNS discovery.

Now, a brief overview of the components in our system: user interface, frontend API, operator, Termproxy, and user environments. The user interface is written with React and JavaScript which communicates with the frontend API and Termproxy to manage challenge environments and connect to them. The frontend API communicates with the Operator to signal that we need a new challenge environment created for a player, and serves the various data requirements for the user interface, like listing challenge sets, challenges, user statistics, etc. The Operator is a layer between the frontend API and Kubernetes, it acts as a method of provisioning isolated environments for players to do their tasks inside of. Termproxy acts as a middleman between the user interface and the user environments themselves. It utilizes gRPC-Web to allow for the web browser to have a native-feeling terminal. Lastly, the user environments are containers running in Kubernetes to provide the player their virtual system and the target systems (if applicable to the challenge).

Lastly, included in the **System Architecture Appendix** are multiple diagrams that describe our architecture, a couple of component diagrams regarding subsystems and a sequence diagram showing the process of the user interface starting and connecting to a challenge environment.

## Subsystems

### III.2.1. User Interface

#### Description

The requirements as set out by our sponsor were to have the lowest barrier-to-entry that was possible. Thus, we targeted the web browser to allow our players to connect to challenge environments running in the cloud.

#### Design Considerations

We needed a modern-looking user interface while also being as quick to develop as possible. We chose to utilize React and a mix of JavaScript and TypeScript to build the necessary user interface components and build a cohesive experience for players to utilize in a challenge.

#### Interface Description

Services Required:

* REST API: Frontend API
* Terminal over gRPC: Termproxy

### III.2.2. Frontend API

#### Description

The Frontend API subsystem provides the REST API that is utilized by the web user interface, and allows it to query challenges and spin-up new environments for those challenges to take place in. It also tracks the user’s progress in the database and will allow them to compare their stats to other players.

#### Design Considerations

The members of the team that are working on this component have experience with Python, thus having a Python-based web framework makes sense. We are utilizing Flask for the web server SQLAlchemy for the SQL ORM to connect to PostgreSQL.

#### Interface Description

Services Provided:

* REST API
  + Service provided to: User Interface
  + Description: The Frontend REST API is used by the user interface to list challenge sets and individual challenges, start an environment for a challenge to take place in, and manage their account.

Services Required:

* PostgreSQL wire protocol: PostgreSQL

### III.2.3. Operator

#### Description

The Operator subsystem is utilized by multiple other components to manage the challenge environments that are created for the players. It is named after the Kubernetes concept of an operator, which manages resources in a Kubernetes cluster by reconciliation of a known config and the actual state of the cluster. It is not only used to create, update, and destroy environments, but it allows the lookup of information on an environment, like the SSH information for Termproxy.

#### Design Considerations

Kubernetes was chosen quite early on to provide isolated environments for players to utilize. A Kubernetes operator is exactly what is needed to manage a cluster for our use cases, and can be expanded to support more backends, like AWS EC2 or Google Compute Engine in the future.

#### Interface Description

Services Provided:

* EnvironmentProvisioningService
  + Service provided to: Frontend API
  + Description: This service is used by the frontend API to request that a new challenge environment be provisioned in Kubernetes.
* EnvironmentLookupService
  + Service provided to: Termproxy
  + Description: This service is used by Termproxy to look up the relevant details for a challenge environment to connect, like SSH service IP and port.

### III.2.4. Termproxy

#### Description

The Termproxy subsystem allows web browsers to connect to a terminal over SSH. This is not generally possible, because JavaScript does not have access to a native socket. Termproxy utilizes gRPC and gRPC-Web to create a bi-directional set of streams that allow us to have a terminal running without a socket connection.

#### Design Considerations

Connecting to a terminal from a web browser allows the barrier of entry to be very low. The Termproxy subsystem was built to provide exactly this service and does it in a very small implementation with Go and gRPC. In the future, it may proxy more than just terminals, like HTTP or other protocols.

#### Interface Description

Services Provided:

* Terminal over gRPC
  + Service provided to: User Interface
  + Description: The user interface utilizes this service to create a bi-directional stream of stdout/stderr/stdin messages and allows the player to use a real Linux shell running inside of a container.

Services Required:

* EnvironmentLookupService: Operator
* SSH: User Environments

### III.2.5. User Environments

#### Description

The user environments subsystem is a collection of isolated environments that are provisioned for users to have access to a shell in which they can use to perform the tasks that relate to a challenge.

#### Design Considerations

One of the goals set out by our sponsor was easy to use and as realistic as possible, so our thought was to use virtual machines or containers, and containers was the obvious choice due to simplicity and low overhead. Our containers are orchestrated by Kubernetes and the images are built with Docker using actual distributions and real penetration testing utilities.

#### Interface Description

Services Provided:

* SSH
  + Service provided to: Termproxy
  + Description: Allows the player to connect to their penimage container inside of their personal challenge environment running in Kubernetes. Utilizes SSH public and private keypairs that are shared between the Operator and the Termproxy subsystems.

### III.2.6. PostgreSQL

#### Description

This subsystem is external but is included to provide a clearer idea of our system’s architecture. It is the PostgreSQL RDBMS running inside of a container on top of Kubernetes.

#### Design Considerations

Our team has varying levels of experience with PostgreSQL, and it was the obvious choice. We could have also considered a NoSQL database, but given the experience, it was better to utilize something that we were already familiar with.

#### Interface Description

Services Provided:

* PostgreSQL wire protocol
  + Service provided to: Frontend API
  + Description: Provides the standard PostgreSQL connection over their wire protocol, allows querying, insertion, updates, etc. in our database.

### III.2.7. Kubernetes

#### Description

This subsystem is external but is included to provide a clearer idea of our system’s architecture. It is Kubernetes, the container orchestration platform.

#### Design Considerations

A member of our team has experience with Kubernetes, another reason to utilize this technology. Not only this, but it gives us the requirements set out by our sponsor, for user environments to be isolated and as real as possible.

#### Interface Description

Services Provided:

* kube-apiserver
  + Service provided to: Operator
  + Description: The kube-apiserver service provides an interface to manage resources in a Kubernetes cluster over HTTPS.

### III.2.8. Auth0

#### Description

This subsystem is external but is included to provide a clearer idea of our system’s architecture. It is Auth0, which is a third-party authentication service that provides free and paid plans to manage users, authentication, and authorization.

#### Design Considerations

At first, we considered doing all user account management by ourselves, but realized that an external service would accelerate the development of this requirement, so we decided to change gears and try out Auth0.

#### Interface Description

Services Provided:

* Auth0 services
  + Service provided to: User Interface, Frontend API, Termproxy
  + Description: The Auth0 service provides several APIs that let our user interface and backend hook into their authentication services and quickly add user accounts to our platform.

# Data Design

## IV.1 PostgreSQL

The Frontend REST API is utilizing PostgreSQL as a backend datastore. We chose PostgreSQL because some members of our team have prior experience with it. We also chose PostgreSQL to futureproof in case we need more advanced database features like relationships between tables and table joins. Currently, challenge sets, challenges, and documentation tables are implemented in our database. The user table and challenge progress tables have not been implemented yet. Refer to the Challenge Set Class Diagram in the **Data Design Appendix**.

### IV.1.1 User

The User table will store relevant user profile information that Auth0 does not handle for us. This could include the user’s name, gender, profile picture, cumulative score, etc.

### IV.1.2 Challenge Set

The Challenge Set table stores the challenge sets that are defined by the challenge archives, which are uploaded through the Frontend API. We utilize a database to speed up the queries against commonly needed information.

### IV.1.3 Challenge

The Challenge table stores the challenges that are defined by the challenge archives that are uploaded through the Frontend API and will have a parent challenge set as a foreign key.

### IV.1.4 User Challenge Progress

This is where user challenge progress will be saved. Current plans are to track which challenges the user has completed. If time permits, we may add a save state property to this table.

### IV.1.5 Documentation

This table stores the documentation for challenge sets and challenges within a set. Documentation is tied to both a challenge set, and particular challenges, as one or more challenges can refer to the same document.

## IV.2 Operator

The Operator utilizes a key-value flat-file database from the bbolt library [1]. We decided to go this route, as the storage requirements for this microservice are very low. Additionally, it is important to note that we do not want to utilize the same database as the Frontend REST API, as due to best practice, individual microservices should not share databases between themselves and other microservices.

### IV.2.1 Environments

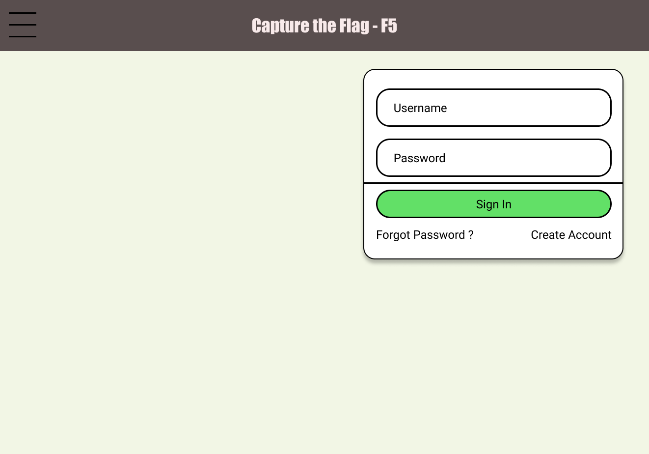
The Environments table will store basic information about running environments and how to manage them. It is a temporary, but persistent store in which the key-value pairs stored will have the same lifetime as the environment that it represents.

### IV.2.2 Challenge

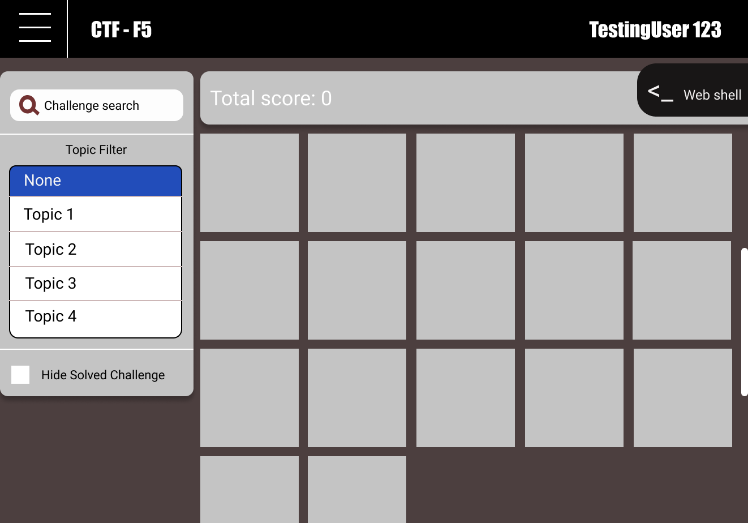
The Challenge table will store information about how to provision certain types of challenges. A challenge may require zero or more containers, and we need to keep a copy of what is required of us to provision on-demand for players.

# User Interface Design

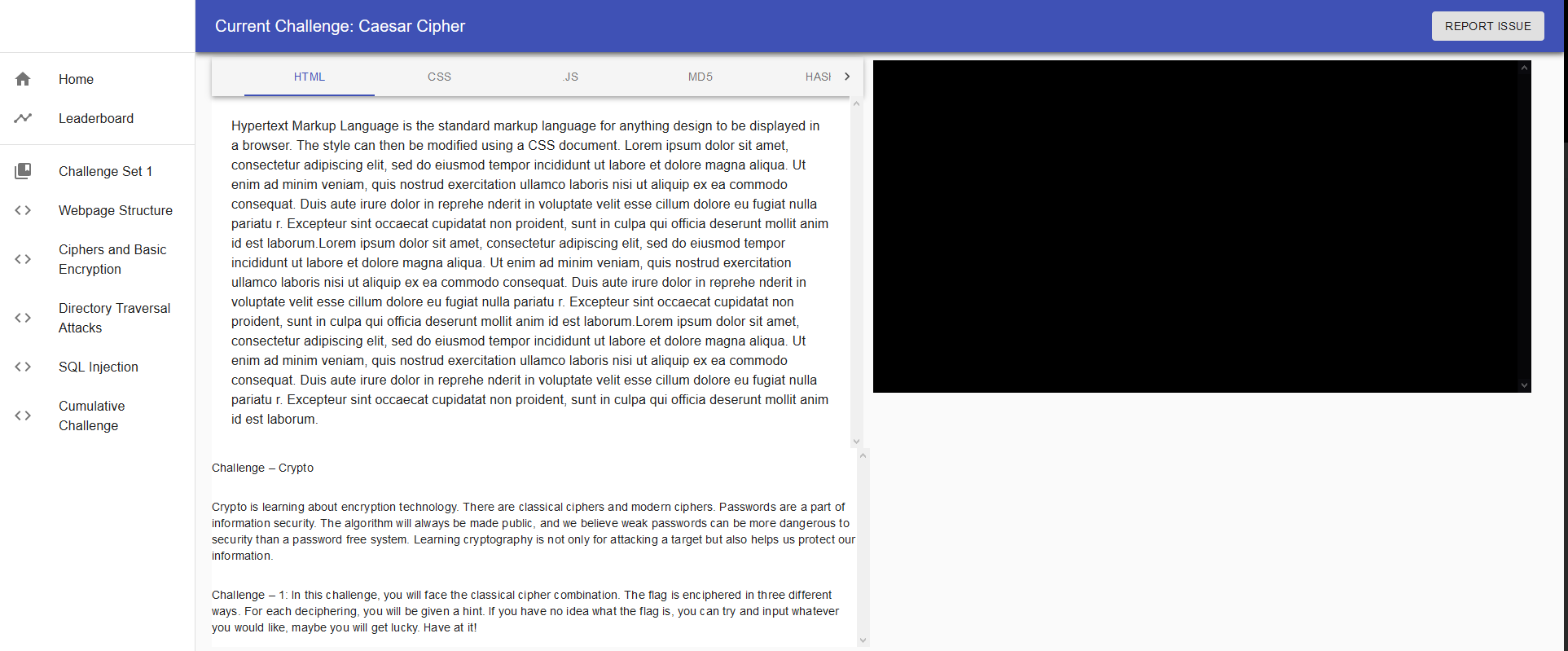
Our project’s user interface is being implemented as a web application. We will be using HTML, CSS, JavaScript and React.js to create the user interface. Users will be presented with a login screen which will also provide the ability to create an account.



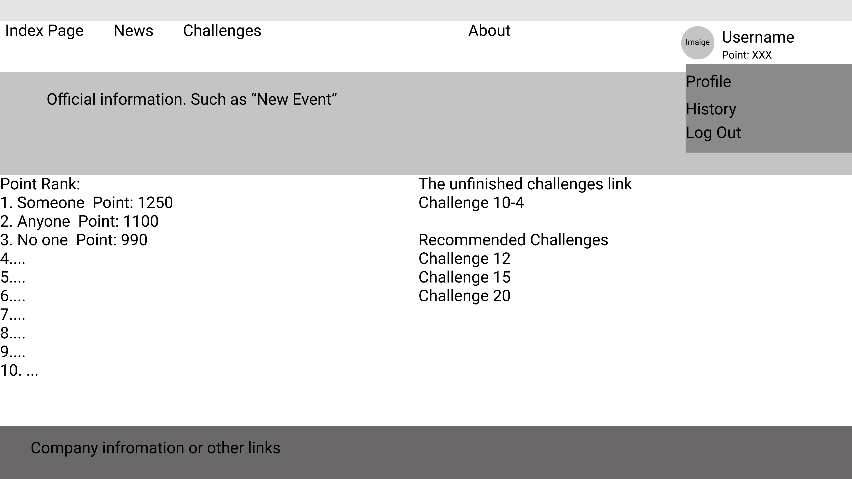
Once logged in, users will be presented with a page that displays the different challenges as sets of cards. They make pick one of the challenges to start by clicking on it.



Once a user has selected a challenge, they will be taken to a challenge screen. In the challenge screen there will be a description of the challenge and a list of navigable documents that are there to help the user overcome the challenge. As it currently stands, the page below and the routing of the challenges is the only image that is of an actual implementation. The others are mockups that we are using to design the actual pages as we go.



After completing a challenge, users will be given points that accumulate as they complete more challenges. If the user is participating in an event where there are multiple users competing there will be a leaderboards page available.



# VI. State of Project

The current state of our project is on track. We have yet to accomplish any of the epics in full but have completed a few user stories and many subtasks. The completed tasks have been highlighted below. To get a more complete picture of where we are in our development process, please view the Epics tables in the **State of Project Appendix**. I have included a list of what we have accomplished for the relevant categories here. What we have not accomplished in the way of user stories and epics is visible in the Appendix where there is a table with a column denoting its state of completion at the end of each user story.

## VI.1. Tasks

### VI.1.1. Completed Tasks

* Implemented basic challenge UI.
  + Challenge Documentation visible in challenges.
  + Terminal able to connect to remote server.
* Implemented page routing between challenge pages.
* Functional terminal implemented in user interface.
* Kubernetes Operator can provision basic challenge environments.
* Design a challenge-set outline.
* Challenge-set 1 outline complete.
* All microservices have been deployed to Kubernetes.
* Docker images are built and live in GitHub Packages.

## VI.2. User Stories

This information is represented in greater detail in the State of Project Appendix.

### VI.2.1. Completed User Stories

* As a user, I want to be able to run this in an online shell environment, so I do not have to install anything on my personal computer.
* As a user, I want challenge documentation to be available so that I can learn about the topics before I apply them in a challenge.
* As a user, I want to be able to report problems to help improve the challenges.

This information is represented in greater detail in the State of Project Appendix. We do not expect to finish any epics by the end of sprint 2.

## VI.3. Epics

### VI.3.1. Completed Epics

* None

Also listed in the **State of Project Appendix** is a table with the expected timeline of all tasks handed out so far. The table was created using Microsoft Project which allowed me to allocate resources to certain tasks and estimate timelines. Below the table is a Gantt chart showing the timeline of all the members on our team and their *expected* contributions to the project by the end of sprint 2.

# Glossary

**Auth0**: Auth0 is a third-party API for adding user account login and registration to a web application. It does not require you to have a database, nor does it require you to securely store passwords. It utilizes JSON Web Tokens as a guarantee that a user has authenticated with them.

**Challenge**: A challenge is a lesson for a particular skill. For example, there may be one challenge for basic ciphers, one for hashes, one for SQL injection, etc.

**Challenge Set**: A challenge set is a collection of one or more related challenges.

**CLI**: Command Line Interface.

**Docker**: Docker is an open-source project that utilizes the concept of namespaces in the Linux kernel to provide secure and lightweight application hosting that is isolated from the rest of the system. Docker can be used to build container images and host them.

**gRPC**: gRPC is an RPC framework that utilizes the code generation of Protocol Buffers (Protobuf) to create a cohesive and simplistic means of machine-to-machine communication.

**JSON Web Token (JWT)**: A JSON Web Token, or JWT, is a means of representing and verifying claims securely through two or more services [2].

**Kubernetes**: Kubernetes is a container orchestration platform that is supported in every popular public cloud provider. It utilizes the CRI interface to allow for any supported platform (Docker, containerd, CRI-O) to host containers.

**Markdown**: Markdown is a standard markup format for creating rich text and is widely used with online tools such as GitHub, Slack, Discord, etc.

**Microservice**: A microservice is a small application that encapsulates a small piece of functionality in a larger system.

**Microservice architecture**: One or more microservices are included to create a single microservice-based system.

**Node**: A Node is a machine running Kubernetes and hosts one or more Pods.

**Pod**: A Pod is a Kubernetes resource that is made up of one or more containers running on a single node.

**Protocol Buffers (Protobuf)**: Protocol Buffers is an open-source library and ecosystem created by Google to allow for simplistic data structures to be defined in a generic language. Protobuf is supported for almost every single programming language, by utilizing code generation.

**UI**: A User Interface where the user can manipulate the program or application.

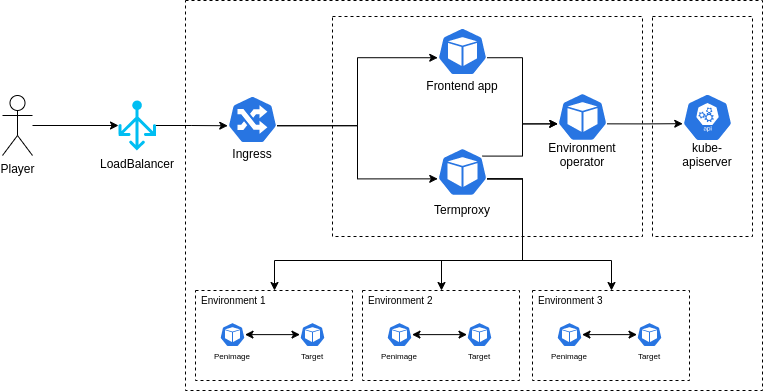
# References

|  |  |
| --- | --- |
| [1] | etcd contributors, "bbolt GitHub repository," [Online]. Available: https://github.com/etcd-io/bbolt. [Accessed 23 March 2021]. |
| [2] | Auth0, "JSON Web Tokens - jwt.io," [Online]. Available: https://jwt.io. [Accessed 2021 30 3]. |

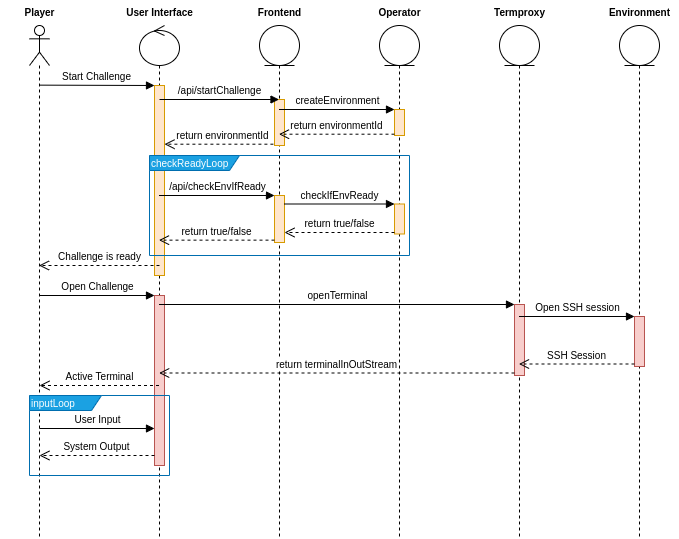
# Appendix

## System Architecture

### Kubernetes Component Diagram



### Challenge Sequence Diagram



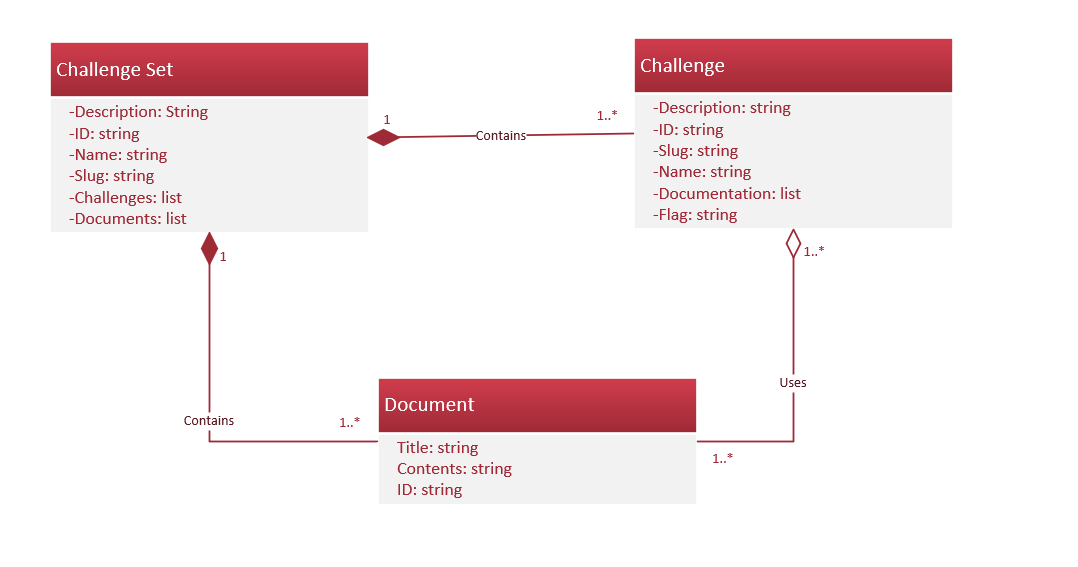
### Architectural Component Diagram

Diagram, schematic

Description automatically generated

## Data Design

### Challenge Set Class Diagram



## State of Project

### Epics

#### Epic 1 - User Interface

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| User Stories | Acceptance Criteria | Story Points | Priority level | Complete |
| As a user, I want to be able to run this in an online shell environment, so I do not have to install anything on my personal computer. | * Application runs in a web browser. * Environment is secure from outside threats and other users using network policies. | 3 | High | Yes |
| As a user, I want to be able to view available challenges so I can select one to attempt. | * Home screen displays available challenges that the user can select from which will spin up a container. | 1 | Medium | In Progress |
| As a user, I want to be able to create an account so that I can track my progress. | * User can create an account that they can log into. * Account keeps track of completed challenges. | 2 | Low | In Progress |
| As a user, I want to be able to modify or delete my account to keep it up to date. | * User can view their public profile and modify it. * User can view their private details and modify them. | 1 | Low | No |
| As a user, I want to be able to view my own statistics to compare myself against others. | * User can view their own statistics. * Users can compare their own statistics against a leaderboard. | 1 | Low | No |
| As a user, I want to be able to filter the available challenges to better find ones that are applicable to me. | * User can easily see the filter controls on the view challenges screen. * User can filter by challenge topic or category. * User can filter by challenge difficulty. | 1 | Low | No |
| As a user, I want to be able to report problems to help improve the challenges. | * User can easily find the “report a problem” button on any screen of the application. * User can easily fill out a title and description to submit with any logs / metrics from the application. | 1 | Low | Yes |

#### Epic 2 - Administrator

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| User Stories | Acceptance Criteria | Story Points | Priority level | Complete |
| As an administrator, I want to be able to set the application up easily. | * Administrators can easily access documentation to get started with the platform. | 1 | High | In Progress |
| As an administrator, I want to have access to documentation for setting the platform up in both a bare metal and a cloud environment. | * Administrators can easily access documentation that shows them how to install in a bare metal environment. * Administrators can easily access documentation that shows them how to install in a cloud environment (e.g., AWS, Azure, GCP). | 2 | High | In Progress |
| As an administrator, I want to have access to documentation to be able to create new challenges. | * Administrators can easily access documentation that shows them how to create and edit challenges and deploy them to the platform. | 2 | Medium | In Progress |
| As an administrator, I want to be able to view system logs so that I can make sure no one is abusing the platform. | * Administrator can view the resource usage of each user. * User can view logs from terminal sessions. | 1 | Low | No |
| As an administrator, I want to be able to view user statistics so I can see if users struggle with a particular challenge. | * Be able to compare the number of attempts with the number of successful attempts. | 1 | Low | No |
| As an administrator, I want to be able to view user reports and feedback so that I can make improvements to the system. | * Be able to view user submitted comments. * Separate user comments by type (e.g., bug report, feature request, other) | 1 | Low | Part 1 complete |

#### Epic 3 – Challenges

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| User Stories | Acceptance Criteria | Story Points | Priority level | Complete |
| As a user, I want challenge documentation to be available so that I can learn about the topics before I apply them in a challenge. | * Users can access readily available documentation that is concise and well-written. * Users can have the documentation open at any point, even with a challenge running. | 3 | High | Yes |
| As a user, I should be able to pick from at least two challenges that cover unique and interesting topics. | * User can pick from at least two different challenges designed by our team. * User can decide between an easier or more difficult topic, and the difficult one may depend on skills learned from the easy one. | 3 | High | In Progress |
| As a developer, I want to be able to save the state a challenge is in, so the user does not have to start from the beginning of each challenge each time they log in. | * Allow the user to perform some sort of state save where their current progress will not be lost if they must do something else in the middle of a challenge. * Could be implemented with checkpoints based on the complexity of the specific challenge. | 3 | Medium | No |
| As a developer, I want the system to provide context-aware hints so that users do not become discourage. | * System can detect if user has been in a state for longer than a pre-determined amount of time and provide them with a hint that is relative to their current progress within the challenge. | 3 | Medium | No |

## Schedule

### Project

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Task Name | Duration | Start | Finish | Predecessors | Resource Names |
| Planning | 1 day | Tue 2/23/21 | Tue 2/23/21 |  | Colby,Ethan,Logan,Richard,Zhaohu |
| UI Design Prototypes | 1 day | Wed 2/24/21 | Wed 2/24/21 | 1 | Colby,Ethan,Logan,Richard,Zhaohu |
| Termproxy prototype | 3 days | Thu 2/25/21 | Mon 3/1/21 | 1 | Logan |
| Set up linter | 1 day | Tue 3/2/21 | Tue 3/2/21 | 1 | Logan |
| Build Docker Image | 1 day | Wed 3/3/21 | Wed 3/3/21 | 3 | Logan |
| Kubernetes Operator | 1 day | Thu 3/4/21 | Thu 3/4/21 | 5 | Logan |
| Write Tests | 1 day | Tue 3/9/21 | Tue 3/9/21 | 6 | Logan |
| Repository Activities | 2 days | Fri 3/5/21 | Mon 3/8/21 | 1 | Logan |
| Kubernetes Deployment | 1 day | Thu 3/11/21 | Thu 3/11/21 | 7,12 | Logan |
| Challenge File Format | 4 days | Mon 3/15/21 | Thu 3/18/21 | 9 | Logan |
| Challenge Design Discussion | 1 day | Thu 2/25/21 | Thu 2/25/21 | 1 | Colby,Ethan,Zhaohu |
| UI implementation prototype | 7 days | Tue 3/2/21 | Wed 3/10/21 | 2,3 | Colby,Richard |
| Design a challenge set outline | 5 days | Fri 2/26/21 | Thu 3/4/21 | 11 | Ethan,Zhaohu |
| Implement Challenge 1 | 3 days | Fri 3/5/21 | Tue 3/9/21 | 13 | Ethan,Zhaohu |
| Document Challenge 1 | 3 days | Wed 3/10/21 | Fri 3/12/21 | 14 | Ethan,Zhaohu |
| Set up UI page routing | 7 days | Thu 3/11/21 | Fri 3/19/21 | 12 | Richard |
| Complete Challenge set 1 | 14 days? | Mon 3/15/21 | Thu 4/1/21 | 13,14,15 | Ethan,Zhaohu |
| Database Diagram of challenges and documentation | 7 days | Thu 3/11/21 | Fri 3/19/21 | 12 | Colby |
| Set up UI page routing | 7 days | Mon 3/22/21 | Tue 3/30/21 | 12,18 | Richard,Colby |
| Project Solution Approach | 3 days | Mon 3/29/21 | Wed 3/31/21 | 1,2,3,11,12,13 | Colby,Logan |
| Link pages with correct data objects in UI | 3 days | Fri 4/2/21 | Mon 4/5/21 | 19,17 | Colby,Richard |
| Import challenge into system | 1 day | Fri 4/2/21 | Fri 4/2/21 | 17 | Logan |
| Build Challenge Command Line Interface | 2 days | Sat 4/3/21 | Mon 4/5/21 | 22 | Logan |

### Gantt Chart

