**Final Report**

**for**

**Tokenizing Employee Rewards: A Blockchain-Based Incentives System**

**Version 1.0**

**Prepared by Daniel Dogbey, Ryan Evans, Rithvik Kalidindi,   
Bennett Quigley, Philip Wallis**

**University of Texas at Dallas**

**May 5, 2023**

**Table of Contents**

[**1. ABSTRACT 1**](#_heading=h.ypa8irnpvoi3)

[Keywords 1](#_heading=h.tmuox0v2byem)

[**2. INTRODUCTION 2**](#_heading=h.oet7gioi5ohq)

[Project Objective 2](#_heading=h.cywryx8vu04q)

[Unique Features and Benefits 2](#_heading=h.hys7o2jteoqs)

[**3. TIMETABLE 3**](#_heading=h.m9e3koc5oinr)

[**4. PROJECT METRICS 4**](#_heading=h.wykdpkvd72fq)

[Project Deliverables 4](#_heading=h.id9rqn28sk6q)

[Use Case Deliverables 4](#_heading=h.dfnru9irmylr)

[**5. IMPLEMENTATION DETAILS 5**](#_heading=h.lv3leaqeial0)

[Database 6](#_heading=h.7qpk1twe9za1)

[Blockchain 7](#_heading=h.xck5wlhp51p0)

[Backend 8](#_heading=h.2winumoo6hti)

[Frontend 8](#_heading=h.hl9bmhd8ome8)

[Deployment 12](#_heading=h.jut5f942ti77)

[**6. IMPACT AND SECURITY 13**](#_heading=h.fb9d5x9fds6l)

[Impact 13](#_heading=h.e9zr61dt3xxp)

[Security 13](#_heading=h.izb060g9sfg)

[**7. INDIVIDUAL ASSESSMENT 14**](#_heading=h.y9mqd6jr1diy)

[**8. ISSUES AND LESSONS LEARNED 16**](#_heading=h.hr5g5hy7aj7f)

[**9. FUTURE WORK 17**](#_heading=h.8lb4ybytuty3)

[**10. ETHICS DISCUSSION 18**](#_heading=h.44xqhlnn7flx)

[**11. SIGNATURES, NAMES and DATE 19**](#_heading=h.5sa6hrtx0u8m)

[Revision History 19](#_heading=h.hzl2nkhdamia)

[**Print Name/Signatures/Date 19**](#_heading=h.3pows0dfx37m)

# 1. ABSTRACT

The purpose of the project is to create a prototype ecosystem, using the Ethereum blockchain to create an ERC20 Token, "TFS Coin", that can be used by the Toyota Financial Services to reward employees. The student team used and created a MySQL database on Amazon Web Services (AWS), Relational Database Service (RDS), two Ethereum smart contracts tested on the Sepolia test network, a NodeJs / ExpressJS backend server, and a ReactJS frontend. The result is an interactive website that the TFS employee can utilize to earn and redeem "TFS Coin" without directly interacting with the blockchain. The website has login authorization methods implemented by JWT, and an interactive rewards store that a user can claim rewards from. The website also includes two methods of earning coins: a daily check-in, and a bug bounty system. The website also includes an admin page where authorized users can modify the rewards list and reward bug bounties.

## Keywords

Blockchain, Ethereum, Fungible Token, MySQL, NodeJS, Express, React, AWS

# 

# 2. INTRODUCTION

In an ever-evolving business landscape, organizations persistently seek novel approaches to bolster employee engagement, productivity, and satisfaction. A groundbreaking solution lies in the development of a blockchain-based rewards system, which offers unparalleled transparency, security, and efficacy. This report delves into the conception of a prototype ecosystem, “TFS Coin,” meticulously crafted for Toyota Financial Services (TFS) as a means to acknowledge and reward their employees’ efforts.

## Project Objective

At the core of this project is the aspiration to devise a secure, user-friendly rewards platform, harnessing the prowess of blockchain to establish a transparent and efficient system for employee recognition at Toyota Financial Services. Accessible through a dynamic website, Toyota employees can effortlessly earn and exchange “TFS Coin” tokens for a range of rewards.

## Unique Features and Benefits

The TFS Coin ecosystem is distinct, as it is meticulously tailored to Toyota Financial Services, integrating bespoke features and functionalities that cater to the company's unique requirements. Among these elements are an interactive rewards store, daily check-in incentives, and a bug bounty system, enticing employees to actively engage with the platform.

A handful of existing solutions offer blockchain-based rewards systems, such as NFT projects that use the length the user has been holding onto a specific NFT, and reward the user with a Fungible Token in which they can exchange for a reward. However, TFS Coin stands apart with its customized features that are suitable for further implementation to existing businesses to implement into their existing employee tracking systems. Moreover, the system was tested on a separate test network on Ethereum called Sepolia, ensuring a secure and controlled testing and implementation environment.

Compared to traditional tracking and rewards systems, the TFS Coin ecosystem boasts a numerous advantages:

1. Transparency: By utilizing blockchain technology, a transparent rewards system is achieved where all transactions are publicly documented and immutable, fostering trust among team members.
2. Security: Blockchain technology and smart contracts safeguard the system, rendering it secure and impervious to tampering or fraud.
3. Customization: The platform's adaptability ensures it meets TFS's needs and can accommodate the company's evolving demands.

In essence, the TFS Coin project seeks to construct a blockchain-based rewards system transcending the limitations of traditional systems, while presenting a customized, captivating, and secure solution tailored for Toyota Financial Services. By leveraging blockchain technology, TFS Coin will offer a transparent and efficient platform to recognize and incentivize employee work, ultimately enhancing engagement, productivity, and satisfaction.

# 3. TIMETABLE

Our main development began on the week of March 6th, after our initial project proposal was given and approved by Toyota Financial Services. This gave us 8 development weeks excluding the spring break, or about 2 months to work on the code. Entire progress can be seen from github commits, and below tables marks some weekly highlights of the development progress.

|  |  |  |
| --- | --- | --- |
| Development Time Table | | |
| Week | Work Done | Delivery Date |
| 1 | AWS RDS Server Created. Initial back-end API design. Database tables made with sample data. | 3/10/2023 |
|  | Spring Break | 3/17/2023 |
| 2 | Created placeholder stored-procedures for development. Front-end development started with a home page demo. | 3/24/2023 |
| 3 | Login API created, frontend updated to redirect the user to the rewards or admin page upon successful login request. Retrieve User Information on frontend. Blockchain Smart Contract deployed on Goerli test network. | 3/31/2023 |
| 4 | Balance from blockchain function completed. Front-end shows balance, and transfer function. | 4/7/2023 |
| 5 | Balance Updates on successful transactions. Transferring tokens between users completed. Status Messages provided to the front-end. AWS EC2 Server created. Contract moved to the Sepolia test network. Blockchain functions created. Login states using JWT on frontend. Rewards viewable from frontend. | 4/14/2023 |
| 6 | Blockchain claim rewards function. JWT Authentication implemented to protect the API. Bug Bounty Functions created. User sign up function created. Time tracker feature implemented in a form of daily check-in button. | 4/21/2023 |
| 7 | Replace backend queries with created stored procedures. Admin bug bounty accept function created. User rewards now include images and dropdown menus for each description. Restructured Frontend to have separate applications. | 4/28/2023 |
| 8 | Completed claiming rewards function. Added code documentations to github. Code deployed on EC2 Server. Project completed. | 5/5/2023 |

# 4. PROJECT METRICS

Multiple deliverables were presented in the project, based on use cases. Outlined deliverables are presented in *Project Deliverables*, and app specific deliverables are in *Use Case Deliverables.*

## Project Deliverables

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Deliverable** | **Recipients** | **Delivery Date** | **Delivery Method** | **Status** | **Comments** |
| Website | TFS | May 6, 2023 | github |  | All deliverables are available in github, in <https://github.com/bquigley1/TFS> |
| Database System | TFS | May 6, 2023 | github |  |
| Blockchain Smart Contracts | TFS | May 6, 2023 | github |  |
| Back-end Server | TFS | May 6, 2023 | github |  |
| Working Product Prototype | TFS | May 6, 2023 | github |  |
| Prototype Code | TFS | May 6, 2023 | github |  |
| Prototype Documentation | TFS | May 6, 2023 | github |  |

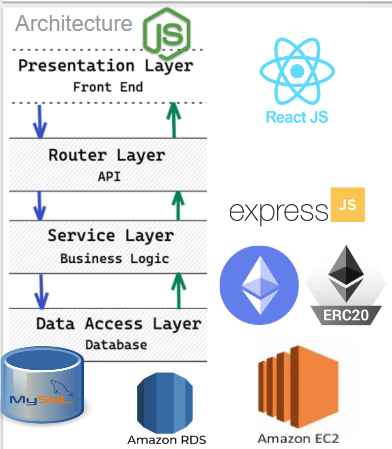
## Use Case Deliverables

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Deliverable** | **Delivery Date** | **Delivery Method** | **Status** | **Comments** |
| Employee Sign In | 3/31/2023 | github |  |  |
| Employee Create Account | 4/21/2023 | github |  |  |
| Employee Claim Reward | 5/5/2023 | github |  | This feature took the longest to implement into the system |
| Employee Transfer Token | 4/14/2023 | github |  |  |
| Time Tracker Application | 4/21/2023 | github |  |  |
| Bug Bounty Application | 4/28/2023 | github |  | Both user/admin sides of the application |
| Admin Update Rewards | 4/28/2023 | github |  | Completed In a form of Adding reward(s) and description(s) |
| Prototype Code | 5/52023 | github |  | All code written is available at <https://github.com/bquigley1/TFS> |
| Prototype Documentation | 5/52023 | github |  | Documentation given in form of README.mb files in github |

## 

# 5. IMPLEMENTATION DETAILS

Our code is divided into 4 layers: Database hosted on AWS RDS, Blockchain hosted on Ethereum’s Sepolia network, Backend API server written in ExpressJS hosted on AWS EC2 server, and finally a Frontend written in ReactJS, also hosted on the same EC2 server.

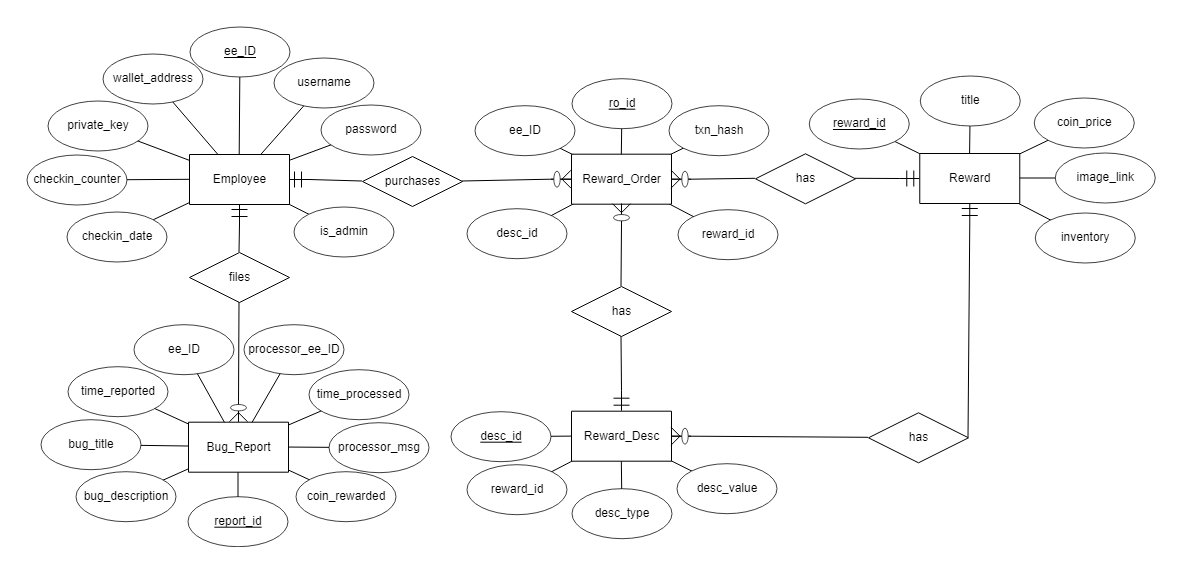


Code Architecture

Our source code documentations can be viewed on the project github, where we have README.md files under each directory that documents available functions and how to get started on running your own instance. <https://github.com/bquigley1/TFS>

|  |  |  |
| --- | --- | --- |
| The preview ecosystem can be viewed at  <http://ec2-3-137-214-39.us-east-2.compute.amazonaws.com:4000/> | | |
| User Sign In:  username: sampleuser1  password: my1password123 | User Sign In:  username: sampleuser2  password: my2password123! | Admin Sign In:  username: sampleadmin  password: securePassword4485! |

## Database



Database Schema

The database has been built in MySQL workbench and deployed to an AWS RDS server. As depicted in the above entity relational diagram, the relational schema is designed to support the functional and nonfunctional requirements of the software system. Furthermore, stored procedures are implemented for backend functions which are optimal for the performance, scalability, and the security of the software system.

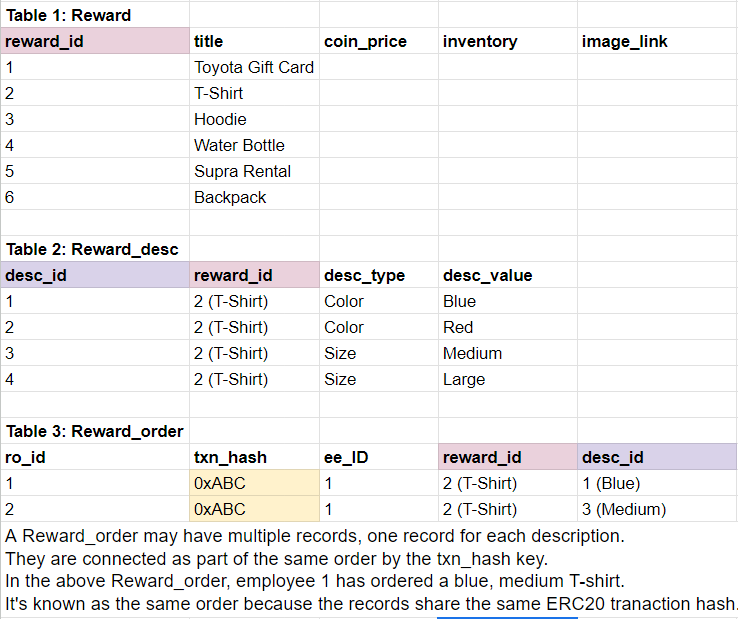
The Employee table stores all information related to employees. It supports logging in, maintaining individual blockchain wallet information, and it maintains the information for the time tracking application as a daily check-in feature. It also keeps track of if a user is an administrator or regular employee.

The Bug\_report table keeps track of all bug report information, and has a relationship with the employee table based on the unique key, ee\_ID (employee ID). It supports the use case of regular employees filing a bug report, and administrators reviewing the bug report before determining an appropriate TFSCoin reward.

The Reward table keeps track of all rewards, their inventory, TFSCoin price, and image links.

The Reward\_Desc table is used to assign qualities to different rewards through the desc\_type/ desc\_value key value pair. For example, desc\_types may be color, size, or length of time, whereas matching desc\_values may be blue, red, green, medium, large, 1 week, etc. The Reward\_desc table references the reward table in order to maintain which description refers to which reward.

Finally, the Reward\_Order table keeps track of which employee purchased which reward, and which qualities they selected with that reward. This is accomplished by references to the primary keys of the Employee table, Reward table, and Reward\_Desc table. It keeps track of the blockchain transaction hash for each order, so that relevant data can be applied to the appropriate use cases, such as determining when a transaction occurred, the total value of the transaction, and the wallet information of the transaction. This table allows for the selection of multiple qualities for each reward by the grouping of multiple Reward\_Order records by using the unique txn\_hash (blockchain transaction hash) as a key on the backend, as demonstrated by the below example tables.



Rewards Tables

## Blockchain

The blockchain consists of two smart contracts deployed on the Sepolia testnet. TFSCoin and TFSRewards. TFSCoin is called by TFSRewards whenever coins need to be distributed or burnt to claim rewards, or when users are being awarded for certain tasks. The reason for the separation of logic, aside from it being a good practice, was due to the nature of contracts being immutable. Since the logic behind the coin itself will not change, and its only nature is to exist, it is important to keep that as a separate smart contract. However, the functionalities of the rewards contract can change, as there may be different requirements for the rewards system.

The rewards contract supports admin features such as, pausing individual rewards, pausing all rewards, adding admins, and adding/removing awards. The coin contract only allows adding admins, which in the case of a new rewards contract being deployed, it will allow transactions from that said contract.

The current architecture allows only transactions from admin wallets, but all admin wallets are approved for sending transactions on users behalfs. Each admin wallet will be funded using Sepolia faucets, in order to have the ETH required to send transactions.

## Backend

The backend is built using the ExpressJS module on a NodeJS server. Utilizing environment variables and the mysql package, we initialize the connection to the database as a pool, then connect to the database. We listen for events on port 3000. We have 6 major components in the API structure; middleware, routes, controllers, services, blockchain, and database.

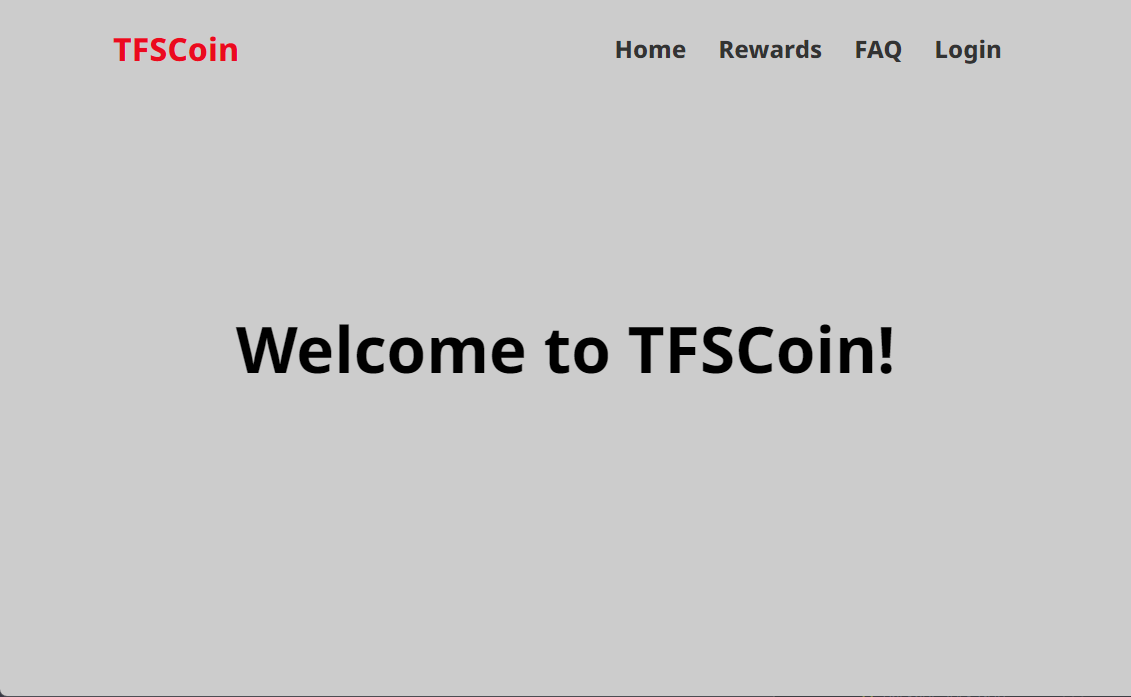
For every API request, we first send the request to an authentication middleware to authenticate the request via JSON Web Tokens and a secret\_key. This process ensures we can keep the API available and secured to only those who have a valid token generated by login. By utilizing a REST API format, we have multiple routes separated by its function: user, rewards, balance, transfer, bounty, and claims. Each of the routes then calls its own controller which responds with a status based on the success values of its services call. In the services component is where the computation is done. From the services, we call either the database or the blockchain to execute the respective functions. The database component is rather simple, which calls the database and returns the value as a Promise given by the stored procedures created in the database. The blockchain component utilizes the web3 package to interact with the blockchain smart contracts and uses the functions in the contract, such as rewarding coins or claiming a reward.

This secure and modular backend server architecture leverages NodeJS / ExpressJS, MySQL, and Web3 modules and environment variables to create a secure, well organized and efficient API server. Utilizing the authentication middleware and clear routing, as well as structure that separates each function to individual components, the resulting server provides both optimal performance and maintainability white providing a seamless experience for the frontend development.

The API documentation in the github provides more details on the created APIs and what each function fulfills.

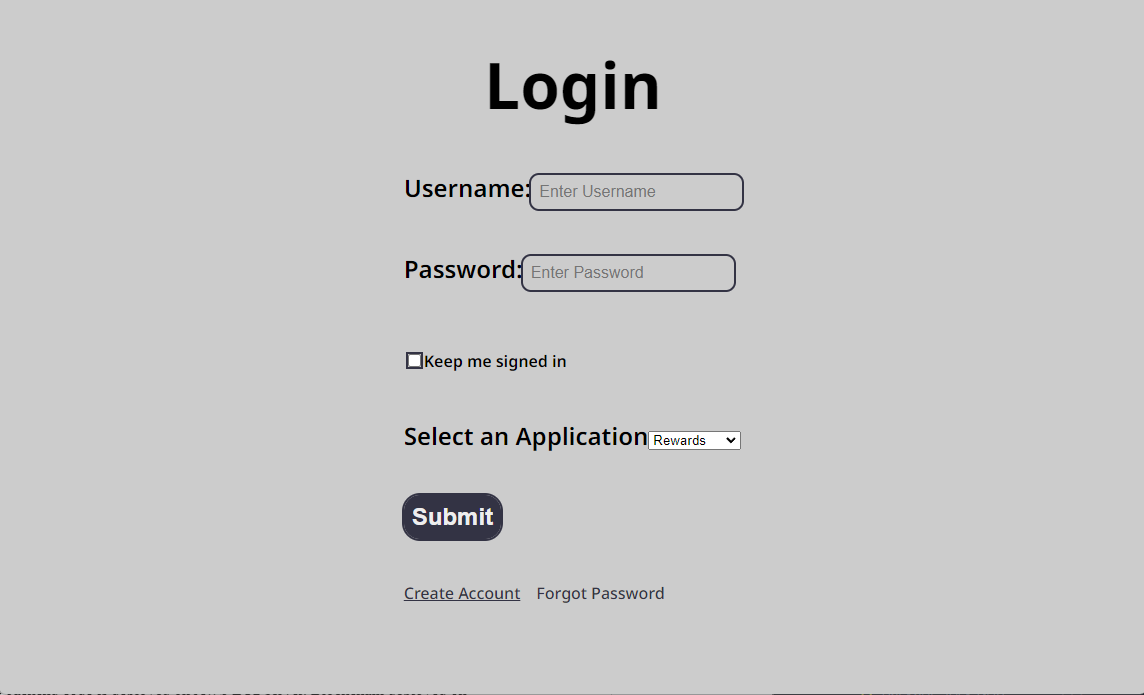
## Frontend

The frontend is built on top of the ReactJS framework and react-router-dom for routing to different pages. Using the create-react-app package, a simple wireframe is generated for further development. Using CSS, the website offers a dynamic and seamless experience for the users on both desktop and mobile platforms. The website has 7 main routes: Home, Login, Signup, Rewards, Bug Bounty, Checkin, and Admin. Home page is a simple page that is displayed when accessing the url, which includes a header that contains the login component.



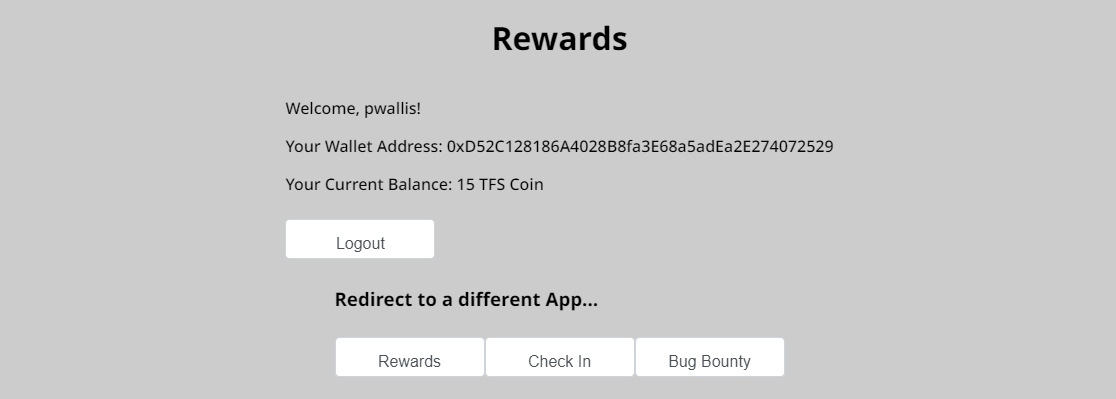
Home Page

Upon clicking login, the user is redirected to the login page where they can sign in using their account or go create an account. The user can select which application they want to use. Rewards, Checkin, or Bug Bounty.



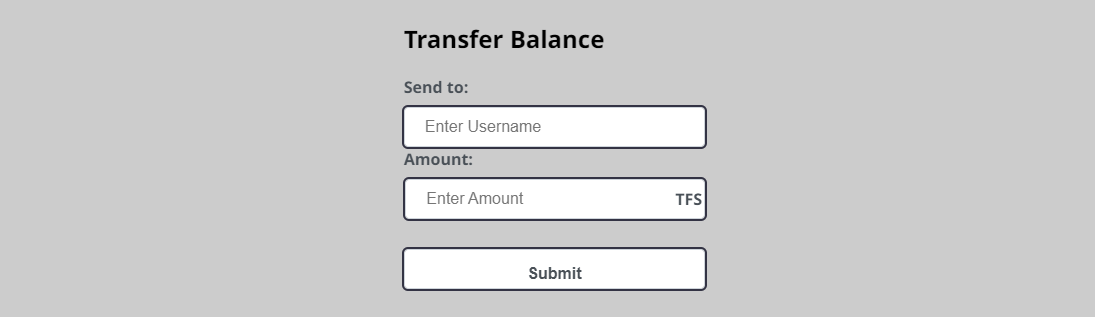
Login Page

Utilizing The login page is the page where the first API call happens as a user. Upon successful login, the JWT will be saved to the localstorage if the user has selected “keep me signed in”. The user is then redirected to the selected application.



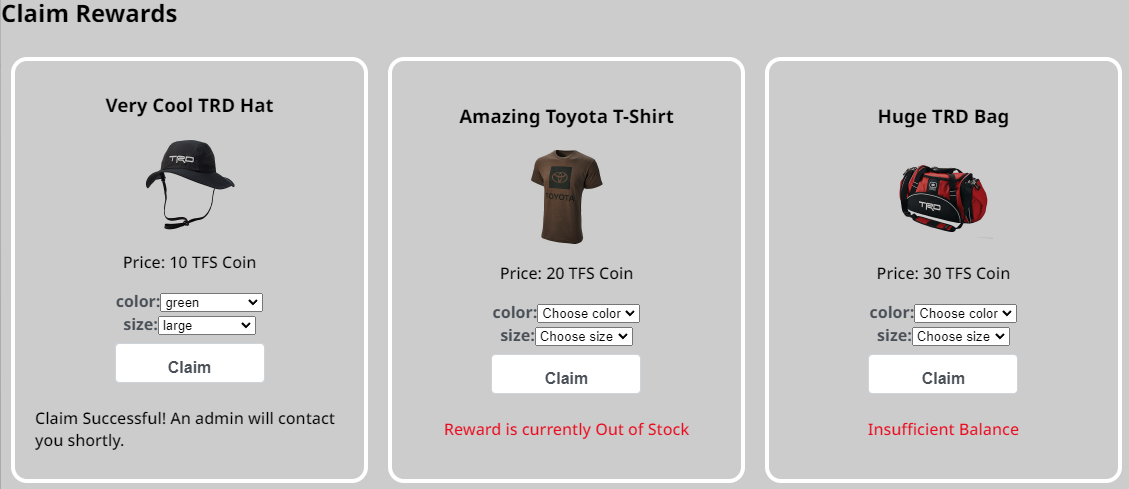
Top of Rewards Page

In the rewards page, it calls the UserInfo component, which displays the username and wallet address that is retrieved from the token created upon login. The UserInfo component also contains the Balance component, which calls an API that returns the balance from the TFSCoin smart contract, and the Logout component that will clear the localstorage and token generated from the browser. On the rewards page is also a Redirect component that lets the user redirect to different applications, whether it be check in or bug bounty.



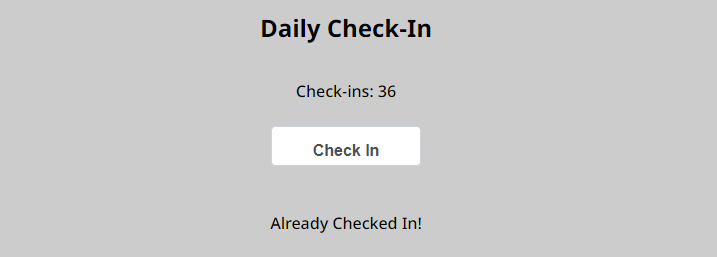
Transfer Balance

In the rewards page, the first interactive component is Transfer Balance. The user can enter a username of a different user, and an amount, to transfer a specific amount of TFS Coins to them. This component calls the Transfer API which interacts with both the database and the blockchain, and returns a success message upon transfer, as well as updating the current balance displayed under UserInfo.



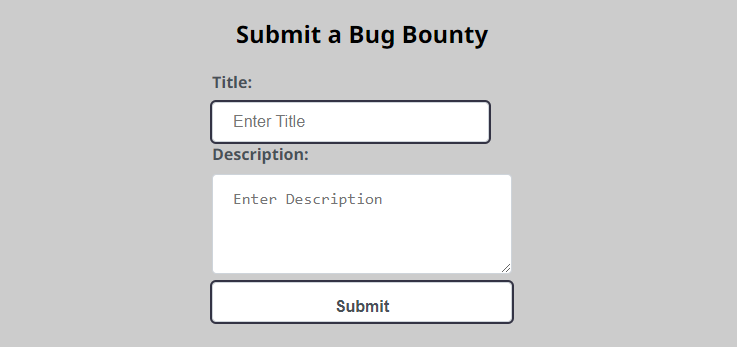
Claim Rewards

The main functionality of the whole prototype ecosystem is presented here. This interacts with all components of the API, database, as well as the blockchain to successfully execute. Error message or a success message is displayed upon clicking the claim button.



Check In page

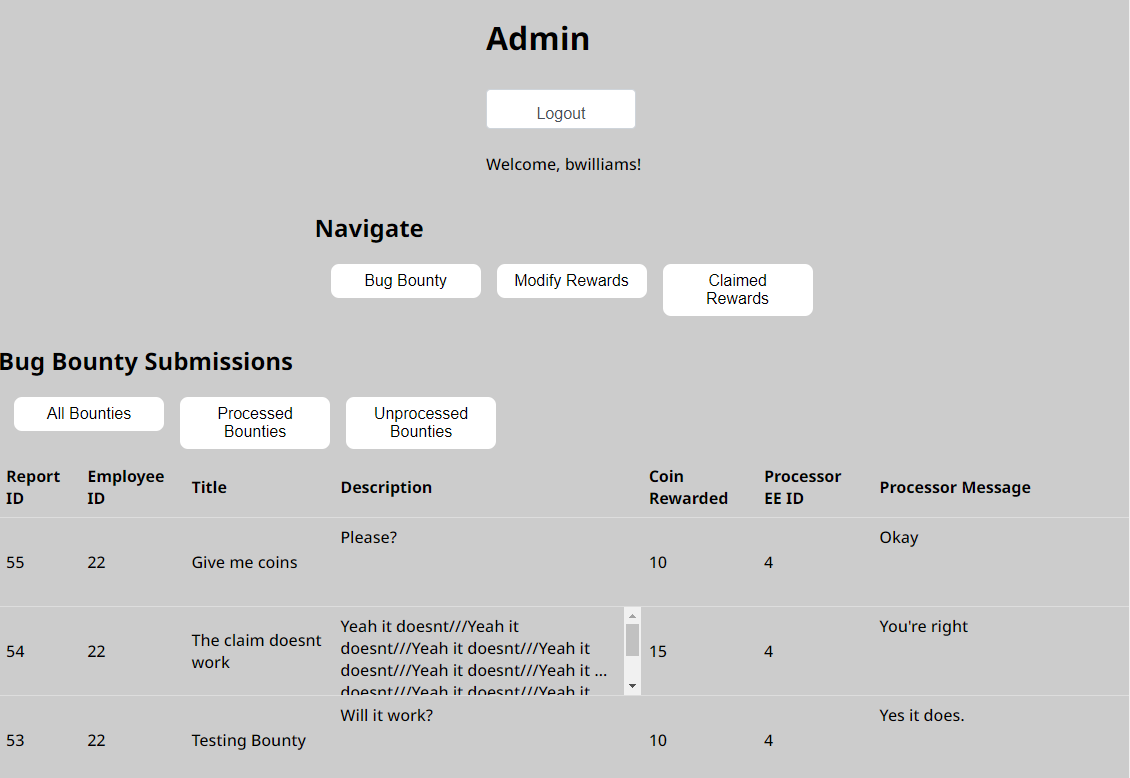
The next application is /checkin . This page displays the UserInfo component, with a single button for checking in. When a user clicks on it, it first checks the database to see if the user has checked in on a given day, and if not, rewards the user with 1 TFS Coin. For every 5 check ins, we reward an extra 4 coins, for a total of 5 coins. For example, on the 24th sign in, the user will receive 1 coin, and on the 25th sign in, the user will receive 5 coins.



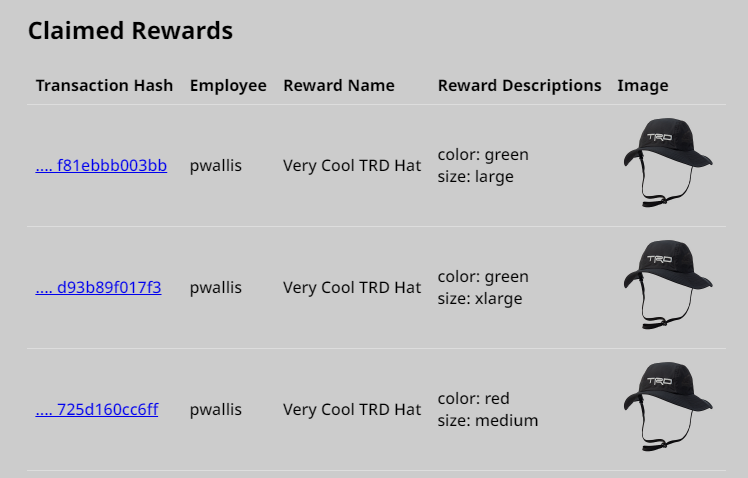
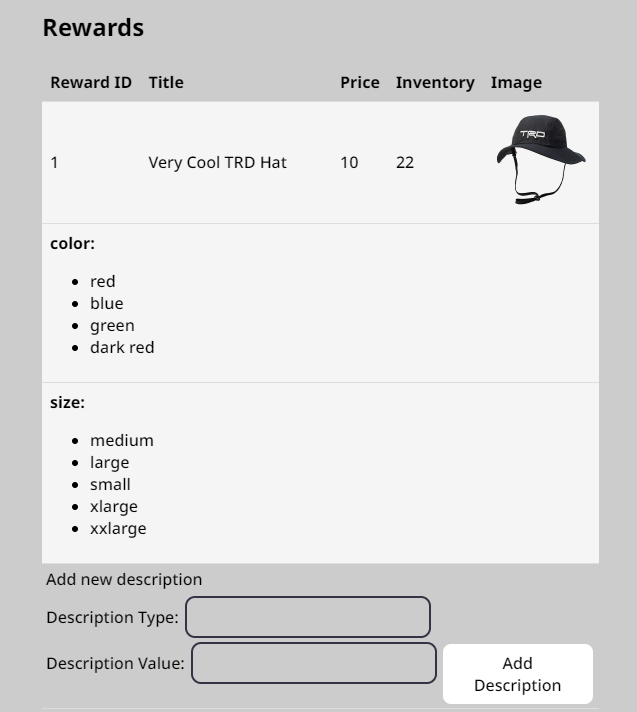
Bug Bounty Page

The final user application is /bugbounty. For bugs, defects, or anything the user deems as a bug can be submitted for a bug bounty. The bug bounty created is sent to the database, which can be processed by an admin at a later time.

Last page provided by the prototype ecosystem is the admin page, that has 3 main functionalities: bug bounty, where the admin can view all, processed, or unprocessed bounties, Modify Rewards, where the admin can view and add a reward or reward description, and lastly Claimed Rewards, where the admin can view the list of rewards claimed by an employee user from the Rewards page.



Bug Bounty Component



Modify Rewards Component, Claim Rewards Component

Upon adding a new reward, the API updates the database, as well as calling a blockchain function that initializes the reward\_id to be available. Claimed Rewards table shows a list of rewards claimed by the employees, which the admin can later contact the user based on the employees username to give them the claimed reward.

## Deployment

The database is hosted on AWS RDS. For the blockchain, we initially decided to go with the Georli testnet because of its Long Term Support, but heard that the support for the test net was ending in 2023, so we decided to change to a different testnet, Sepolia. Functionally, it offers the same system, so transitioning to the Sepolia testnet was not an issue. The frontend and backend code is deployed on an AWS EC2 instance, which differs from our initial plan of using AWS Amplify and Lambda. We decided to make this change due to how our students had experience running EC2 instances before, and setting up the server was similar to the development environment.

# 6. IMPACT AND SECURITY

The prototype ecosystem both creates positive impacts, while keeping the data secure for the users.

## Impact

The implementation of the TFS Coin project is poised to generate positive impacts for its users, whether it be the employees or the employers. By adopting a blockchain-based rewards system, marked by heightened employee engagement, productivity, and satisfaction. This, in turn, will contribute to the company's sustained growth and profitability, solidifying its position as a leader in the financial services industry.

For the employees, the TFS Coin ecosystem will foster a sense of fairness and trust, thanks to the transparency and security afforded by blockchain technology. This will not only boost morale, but also encourage collaboration and innovation, as team members feel valued and recognized for their efforts. As a result, TFS will benefit from increased employee retention and an enhanced reputation as an employer of choice.

At an external level, the successful implementation of the TFS Coin project could inspire other businesses to explore similar blockchain-based solutions, spurring technological advancements and creating a more dynamic, innovative ecosystem. This would contribute to the development of local and regional economies, as well as strengthen the global competitiveness of the area. Ultimately, the TFS Coin project stands as a testament to the transformative potential of blockchain technology, paving the way for a new era of transparent, secure, and efficient employee rewards systems.

## Security

Blockchain technology aids the security of this project in multiple ways. Firstly, the use of blockchain technology creates a decentralized and secure ledger for the TFS Coin transactions, making it almost impossible to tamper with the records. Additionally, the blockchain's immutability feature ensures that once a transaction is recorded, it cannot be changed or deleted, providing an extra layer of security for the TFS Coin ecosystem.

The use of login authorization methods implemented by JSON Web Tokens (JWT) helps to authenticate users and ensure that only authorized users have access to the website's features. The JWT also allows the API to be protected and accessible only to the authorized employees.

SQL Injection attacks are prevented by using parameterized stored procedures. Injection attacks occur when malicious code is injected into an SQL statement. With parameterized stored procedures, the user input is treated as a variable, rather than as part of the SQL statement itself, which makes it much harder for an attacker to inject malicious code into the SQL statement. The database server automatically checks and validates the user input before using it in the SQL statement, preventing any SQL injection attempts.

# 7. INDIVIDUAL ASSESSMENT

This section highlights each team member’s roles and contributions to the project.

|  |
| --- |
| **Daniel Dogbey** |
| I was responsible for the creation of the required smart contracts on the blockchain. This project needed a smart contract for the rewards system and one for the actual token itself, so I was sure to make sure those contracts have the proper connections to each other. I also contributed to the backend development in terms of connecting the blockchain to our backend. Every time I made a new deployment of the smart contracts, which was about every week, I needed to update the contract ABIs we have stored as well as the addresses for said contracts to reflect the changes. Adding admin addresses and configuring the contract for use was my responsibility as well. Additionally, I provided scripts, with documentation, to help interact with the blockchain and gave guidance on further interactions. |

|  |
| --- |
| **Ryan Evans** |
| I was responsible for the database management system, database design, and developing stored procedures for the software system. Throughout the project, I have iteratively made changes to the database design to meet the functional and nonfunctional requirements of the software system, while also communicating the database architecture with backend developers to meet their needs. Furthermore, I created and uploaded to the repository documentation in the form of entity relational diagrams, example tables, and stored procedures. This work was completed on a weekly basis from March 6th until April 28th, 2023. In addition, from February 3rd to March 6th, I contributed to requirements gathering, writing use cases, and creating sequence diagrams to present the software system’s initial architectural design. All of my work was delivered before May 3rd as part of the final software system product. |

|  |
| --- |
| **Rithvik Kalidindi** |
| My role on the team was mainly on the blockchain and design side of the project. I helped design the blockchain architecture and contributed to discussions regarding the overall prototype design. Towards the beginning of the project, I was also active in defining functional and non-functional requirements. Later on, I would help troubleshoot changes to our prototype design based on feedback given by our sponsor. Towards the end of the project, I was tasked with designing our presentation slide, poster, and script. |

|  |
| --- |
| **Bennett Quigley** |
| I began by designing the project's structure, which included the frontend, an Express backend, a database, and the integration of all components. To facilitate collaboration, I set up the GitHub repository and created the backend framework, leveraging my previous experience in this area. I guided the group towards the most effective approach for our project, inspired by Coinbase's master wallet strategy and ensuring that the front-end user didn't interact directly with the blockchain. Throughout the process, I addressed Toyota’s inquiries and sought clarifications when needed. I also tackled backend issues by troubleshooting and refining the code. Collaboratively, I worked with Ryan to establish database procedures and strategize our approach, while also assisting Philip in understanding how to utilize the backend responses effectively. |

|  |
| --- |
| **Philip Wallis** |
| The main role that I was assigned was the development of backend and frontend parts of the ecosystem. Because I had no experience working with creating servers before, with the initial structure generated by Bennett, I created about half of the APIs in the server. During the API generation, I also helped Daniel and Ryan update the blockchain/database by giving suggestions or direct inputs for the frontend needs. Also working with Daniel and Ryan, as I was developing the frontend, communicated my needs with them for the proper blockchain functions and stored procedures that returns the right queries back to the frontend. I developed the entire frontend website, from setting it up, to connecting the backend APIs to the frontend. I also launched the AWS RDS server for the team’s use, as well as the AWS EC2 server for backend and frontend deployment. When I needed specific work done by other components of the project, I contacted the responsible person to work with them, occasionally updating them myself, then notifying and discussing the changes with them. All works done by me can be viewed in the github repository’s commits, done before the final deadline of May 5th. I also created documentation for the frontend, as well as the backend documentation on the github repository. |

# 8. ISSUES AND LESSONS LEARNED

The database systems course included the creation of an entity relational diagram for the purpose of designing the database schema and communicating this design as documentation for other team members to use while programming the backend. A major item learned during this project was designing the database schema iteratively to suit backend needs as development progressed. That is, alternative database schemas may be needed at different stages of development.

The inclusion of a blockchain system with a rewards service showed the necessary components for an intertwined system. The connection of the blockchain to the backend and database were required for the creation of the project, and the team had to ensure a consistent state across all said components. Failure to do so could result in erroneous behavior in the system. It was also learned that access control was necessary in order to prevent exploitation of the system. A key issue we initially had was being able to fund user wallets with ETH to perform all their transactions without giving the user an extra complex step to complete that. As mentioned in section 5, we managed to tackle that and keep the user experience seamless.

An issue that arose during the development of the backend server was how we wanted to structure our API. Since the fully stacked application requires so many functions, adding APIs were built on top of what we created in the first place. Half way during the development, we had to restructure our backend to organize the code in a way that would make it easier for modifications.

Another Major concept presented in the project was frontend development. UTD offers only two courses that touch on the subject of any frontend development, which have a very competitive signups, making it very inaccessible for students to take those courses. Developing the frontend presented a steep learning curve, as the team had minimal experience interacting with backend servers or using the ReactJS framework. This challenge underscored the importance of incorporating frontend development into the broader educational curriculum.

An important lesson taught by the TFS team is to utilize github and its review functions. Although us as students used a platform to communicate with each other, there is no way for the TFS team to know who has been doing what aspects, and whether or not the code will execute on their end. A lesson taught is to utilize the pull requests to do code reviews to make sure the code works on all parties of the development, as well as have versioning available to keep track of any progress made, or in case of an issue, have the ability to revert back to a working version of the code.

# 9. FUTURE WORK

There are several potential developments to the database worth consideration. To begin with, a table in the database for each type of reward in conjunction with proper backend implementation may better allow for the implementation of different types of rewards needing different qualities associated with them for the user to select between. This information would be stored for the user as a single reward order record. This potential upgrade may make it easier to keep exact track of and perform operations on user’s orders, while also making it easier to engineer a dynamic front-end for the rewards application, thereby better supporting and acting as an extension point for future additions to the software system. Moreover, image links stored in the database to keep track of reward images could be replaced with utilizing a custom blob storage server, and using links from our own server, rather than grabbing images used elsewhere.

Another part that we believe could use an improvement is with the individual APIs created. A feature that we decided to pass on is searching a user by name, so the employees don’t have to know the usernames of the other employees. Functions like editing or removing a reward was another function that we skipped out on due to time constraints.

There are many things in the frontend that could have been better. Styling is a main challenge that takes a while given that most students had no experience with CSS styling. Not only that, creating different components, organizing code, calling the api in the right place, and much more could have been improved, given some time.

The interactions with the blockchain could be facilitated with multiple wallets to support no downtime. Currently, there is one wallet that manually needs to be filled with ETH, once that wallet runs out of ETH, the whole rewards system will not work. With our estimations, the current setup should be able to support the current volume of transactions, but could change depending on the amount of active users.

Our daily check in function could have a lot of improvements. Instead of a simple daily check in button, an implementation with work tracking softwares like Jira could have been implemented to reward the employees based on actual work rather than a simple check in every day.

Overall, there are many aspects of the program that could be improved with more work.

# 

# 

# 10. ETHICS DISCUSSION

The use of blockchain technology in the context of this software system prototype is worthy of discussion due to its public nature of Ethereum wallets and transactions, because blockchain systems are not anonymous, but rather pseudo-anonymous; it will be possible for potential ‘attackers’ to gain knowledge of a user's wallet hash identity, thereby gaining the ability to track how much TFSCoin they own, when that TFSCoin is sent to other wallets, and which wallets it is sent to. As a result, there are ethical concerns regarding the privacy of user holdings and transactions.

Regarding the development of the software system, information relating to the architecture, technology, or code from third-party sources was always verified and double checked prior to implementation. Furthermore, any code borrowed or reused from third-party sources was properly documented and cited in accordance with ethical practices. That is, the purpose of the code was always aligned with our ethical principles, and any potential ethical concerns were thoroughly considered and addressed prior to implementation. In conclusion, we prioritize ethical and responsible development practices in all aspects of our work.

Artificial Intelligence softwares such as ChatGPT and Github Copilot were at times used to aid research and development for the software system. No code was directly copied from AI; however, we explored development suggestions, alternative design patterns, and other ideas. Any AI generated information was always cross referenced and verified by legitimate sources before implementation. As a result, the exploration and research stages of the development process were expedited by the power of querying the AI for a breadth of information, ideas, and materials. It is important to note that, although the AIs were trained on public data, there is a possibility that it also trained on private data, which may have been suggested to the students in the development process. Since there is no code copied from the AI generated materials, there will not be any legal issues or ethical concerns; however, it is possible that the structure of some parts of our software system may be comparable to what has already been implemented in other software projects used as training data for the large language model.

# 11. SIGNATURES, NAMES and DATE

## Revision History

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Date** | **Reason for Changes** | **Version** |
| Students | 05/03/2023 | Initial draft | 1.0 |
|  |  |  |  |

## Print Name/Signatures/Date

|  |  |
| --- | --- |
| Student: Daniel Dogbey | Student: Ryan Evans |
| Student: Rithvik Kalidindi | Student: Bennett Quigley |
| Student: Philip Wallis | Faculty Advisor: Shiyi Wei |
| Brandi Barbour  Company Mentor: Brandi Barbour | Company Mentor: Mark Nacionales |