



Google Summer of Code

Proposal - Google Summer of Code 2023

By
Swapnil Shinde



Mentors

Donnie (Lead Mentor)

Aryan Ranjan (Flutter And Django Mentor)

Personal Info

Name: Swapnil Shinde

Email: swapnilshinde9382@gmail.com

TimeZone: India (UTC +5.30)

GitHub: [AtmegaBuzz](#)

Linkedin: [Swapnil shinde](#)

Slack: swapnil shinde

University Info

University: Savitribai Phule Pune University

Current Year: 2nd Year (Expected Graduation : 2025)

Degree: Bachelor of Engineering (BE) in Computer Science

ABOUT ME

I am currently pursuing a Bachelor of Engineering in Computer Science at Savitribai Phule Pune University in India. During my studies, I have gained experience in various programming languages, including Python, which I learned in my first year of college. I then went on to specialize in backend programming and became proficient in frameworks such as Flask and Django. I also have a good understanding of front-end programming using JavaScript libraries such as React, Next, and React-Native.

In addition to my programming skills, I am also familiar with unit testing Python code using pytest and Selenium. I am passionate about learning new technologies and have a particular interest in backend programming, cybersecurity, and low-level system concepts.

I have a basic understanding of Blockchain technologies and had the opportunity to work with smart contracts during my participation in hackathons. Furthermore, I also had the opportunity to work as MLH Fellowship's prep fellow, where I learned the fundamentals of contributing to open source projects.

STATEMENT OF MOTIVATION

I'm choosing OWASP for a particular reason. The majority of the tools I used when I first began learning about cybersecurity ZAP, Top 10, etc were from OWASP, and OWASP Top 10 helped me understand and learn about the top 10 web vulnerabilities. I believe that by making a contribution to OWASP I will be helping in providing free and open source cyber security tools which will make the users more protected towards cyber security attacks.

I chose BLT Project because I like the overall idea of reporting bug's found on the internet and anyone could report it by tagging the organizations which will be incentivized.

BLT aligns with my Stack i.e Django, Javascript, Blockchain. I also have a bit of understanding in blockchain (Bitcoin-core) and this project will help me

improve my skills on the stack listed above.

I have a deep appreciation for the BLT community and its mentors. After being in the community for a while, I had an exceptional experience with the BLT mentors who not only provided me with valuable assistance but also served as my mentors whenever I faced challenges. Being a part of the BLT community has been incredibly fulfilling, and I would relish the opportunity to continue to contribute and help in growing it..

Related Experience

BookCab: Book Cab is a web app that optimizes rideshare bookings by finding the shortest path for shared rides with multiple destinations using an algorithm. Its Ethereum-based payment system is backed by a smart contract deployed on a testnet. The app's algorithm performance will be studied for practical use.

Html, Css, Js, Flask, Ethereum, Solidity

Vault-For-All: Vault-for-all is an Ethereum-based wallet that utilizes Web3 push protocol and Alexa Skills to notify users via Alexa whenever they receive payments from entities. It enables merchants to transition to a voice-based payment notification system on decentralized cryptocurrency and award them NFT's.

React.js, Web3 Push Protocol, FileCoin NFT's

WeebServer: A Simple implementation of a Static Web-Server from scratch. being a simple implementation it can handle routes and status code.

C++, C, make, Socket Programming

Port Scanner: This is a Rust-based multi-threaded port scanner that sends TCP ACK requests to ports and outputs in the terminal if they're open. It employs multithreading to scan multiple ports at the same time.

Rust, TCP, Sockets, MultiThreading

CONTRIBUTIONS IN OWASP

Over the period of 1 year I am actively contributing to owasp projects and actively involved in the community. During this period I have contributed to projects **BLT, PyGoat and Threat-Dragon**.

In Project BLT in total I have :

- [64 merged BLT pr's](#)
- [19 solved BLT issues](#)
- [15 raised BLT issues](#)

Project Details

Proposed Duration: 350 hours

Project Size: Large

The current BLT website has some bugs and the user interface (UI) is not polished. To address this issue, I am proposing to implementing a [New Designs \(#696\)](#) using Tailwind CSS. The plan is to take inspiration from the Reddit UI and combine it with the old BLT designs to create a modern and clean look. In frontend I am implementing new designs and features for **Start Bughunt Page** and **Report A Bug Page**.

In addition to improving the front-end design, I will also be working on the back-end by

1. creating new APIs and integrating an API rate limiter to improve the website's performance and security. **Todo**
2. Implementing a Bacon coin over POA consensus, which is a type of cryptocurrency that will be used to reward users for their contributions to the BLT community. **Todo**
3. To manage the distribution of the Bacon coin rewards, a smart contract will be created and deployed on our BACON MINNET. In this case, the smart contract will automatically distribute the rewards to users based on their participation and contributions to the BLT community. **Todo**
4. Implementing a private issue reporting feature which will enable the reporter to directly report the bug to the Company to which the domain belongs, This way the company can verify and fix the potential issue/bug before publicly displaying the reported bug. **Todo**

5. Implementing a Company Dashboard which will make it easier to manage company roles and domains, start Bughunts, verify bugs, independently reward testers and visualize the company stats. **Todo**

Overall, this plan aims to improve the BLT website's functionality, performance, and user experience while also incentivizing users to contribute to the community.

BLT Frontend

BUGHUNT

- **Start Bughunt Page**

Implement a new Bug Hunt page using **Tailwind CSS** to encourage community involvement in identifying and fixing any issues.

Two new feature buttons will be added in the page which will be integrated to another feature of launching private bug hunts and Anonymous Hunts.

1. "Private Hunt" checkbox will be added to make all the bugs reported private by default. **Todo**
2. "Anonymous Hunts" checkbox will be added which will hide the details of the company and the user who started the bug hunt from the Hunt detail page. **Todo**

Total fields: domain, hunt name, hunt description(markdown), hunt assets, prize pool, hunt logo, private hunt, anonymous hunt, recaptcha

To ensure a seamless transition, I intend to revamp the existing Bug Hunt page with a more refined user interface and a sophisticated background, which will enhance the overall aesthetic appeal of our website. **Todo**

The Designs given below are reference examples. The actual implemented design would be created using these references.

START BUGHUNT

 Enter Website Domain

 Bug Hunt Name

 Upload Bughunt assets

Browse

B

i

⌂

↶

<c>

A^x

🔍

↵

≡

≡

99

📄

📁

🖼️

🎥

Markdown Mode

Text (optional)

Describe Bug Hunt

+ OC

+ Spoiler

+ NSFW

🏷️ Flair ▾

Grand Prize:

Max amout awarded to winner

\$5,300

Upload Logo

No files Chosen

☒ Private Bug Hunt

☐ Anonymous Bug Hunt

☐ I'm not a Robot



START HUNT

START A BUGHUNT

Enter URL, APP name or GPS location of issue

Name of the bughunt

<http://example.com>

No file chosen

Flea Plan \$9.00 / Month

Grand Prize:
Max amount awarded to winner

\$5,300

- **List Bughunt Page**

The List Bughunt page is a web page that will display information about all ongoing bughunt programs. The page will have fields for the domain, bughunt name, description, prize, start and end dates, and a button to participate in each bug. Users will also be able to filter Bug Hunts by status, whether they are ongoing or finished, using a dropdown menu.

To enable users to filter the Bug Hunts, a dropdown menu will be added that allows users to select either **ONGOING/FINISHED** Bug Hunts. This will update the table to display only the relevant Bug Hunts.

Finally, A Button to participate in each bug will be added, which will redirect the user to the Report Bug page. The button will autofill the domain with the bughunt domain to make it easier for users to report bugs.

The page will be designed using Tailwind CSS, a utility-first CSS framework. This will allow for easy styling, customization and maintaining of the page's. The UI of the page will be designed to be visually appealing and user-friendly, with a clean and modern design.

Overall, the List Bughunt page will provide a user-friendly interface for users to view and participate in ongoing bughunt programs.

Report A Bug Page

- UI Revamp
- Allow For anonymous Bug-Reports
- Report Privately

We have identified several limitations in our current "Report a Bug" page that we believe could be improved upon.

Firstly, our current page does not provide users with a convenient way to describe their bug using markdown files or images within the text. This can result in a less than ideal bug report, potentially leading to delays in resolving the issue.

Additionally, we acknowledge that some users may prefer to report their bugs **anonymously** in order to protect their identity. Our current page does not offer this option.

Furthermore, we do not currently have a **"Report Privately"** feature, which

would allow users to report a bug that would only be seen by our company until it is fixed, at which point it could be made public.

To address these issues, I propose a redesign of our "Report a Bug" page that will include a user-friendly interface, improved accessibility, and support for markdown files and images. I will also incorporate the ability to report bugs anonymously and privately, which will increase user trust in our platform.

To achieve this redesign, I plan to create a new Figma design which includes all the features listed above with a clean and modern redesign to this page, which includes fields like:

1. Domain Input field with check for duplicate button.
2. A Markdown Text-Field for Bug report(Description) this can be achieved via the [Django-Markdown](#) package.

```
pip install django-markdown
```

```
from django.db import models
from markdownx.models import MarkdownxField

class Issue(models.Model):
    **previous Issue fields
    content = MarkdownxField()

    def __str__(self):
        return self.title
```

3. Adding a checkbox button for private issue reporting. If checked the reported issue will be private and can only be seen by the Company which the reported domain belongs to, after verifying and fixing the company can make it public.

4. Adding a checkbox for Anonymous issue reporting, this will allow bug reporters to hide their identity while reporting bugs to them as well as maintaining a BLT user profile.

5. Adding a New version of recaptcha to make the backend more safe towards bot activity. For this I intend to use [Django-Recaptcha](#).

Given below is a wireframe design that will give a better understanding of the look and structure of the page and is not a final implementation of the design for this page.

The wireframe design for the 'REPORT A BUG' form is contained within a rounded rectangle. At the top center is the title 'REPORT A BUG' in a large, bold font, with 'REPORT' in black and 'A BUG' in red. Below the title is a light gray input field containing the URL 'https://testdomain.com/bugfound'. To the right of this field is a red button with white text that says 'Check Duplicates'. Below the URL field is a text editor interface. It features a toolbar with icons for bold (B), italic (i), link, unlink, code (<>), text color (A^), background color, bulleted list, numbered list, quote, indent, outdent, table, image, and video. On the right side of the toolbar is a blue link labeled 'Markdown Mode'. The text area below the toolbar contains the placeholder text 'Text (optional)' and the centered prompt 'Describe Issue'. At the bottom of the text editor are four buttons: '+ OC', '+ Spoiler', '+ NSFW', and a 'Flair' button with a dropdown arrow. Below the text editor is a gray tab bar with a hamburger menu icon on the left, the text 'General', and a downward-pointing triangle on the right. Under the tab bar are three options: 'Add Screenshots' (with a subtext 'No files Chosen'), 'Report Privately' (with a selected radio button), and 'Report Anonymously' (with an unselected radio button). Below these options is a reCAPTCHA widget showing the text 'I'm not a Robot' and the reCAPTCHA logo. At the bottom center is a large red button with white text that says 'REPORT'.

BLT Backend

Backend part will be implemented in the following steps:

- APIs
 - Create new APIs
 - Integrate API rate limiter
- Implementing Bacon coin over POA consensus.
- Creating a smart contract to reward the users.
- Implementing Private Issue reporting.
- Implementing Company Dashboard.

API's

1. API'S FOR BLT FLUTTER

a. INVITE A FRIEND REWARD

This API allows authenticated users to invite their friends to join the application. The API requires an access token in the Authorization header for authentication.

The friend_email field in the request body is mandatory and should contain the email address of the friend to be invited. The message field is optional and can be used by the user to add a personalized message to the invitation.

The API returns a JSON response with a success boolean indicating whether the invitation was sent successfully or not. If the invitation was sent successfully, the response also includes a message field with a success message. If there was an error sending the invitation, the response will include a message describing the error.

A token will be generated for a user which will be attached to the

email link. If any user registers with that email link on bugheist then the token will be decrypted to get the Origin User and reward them.

HTTP POST Endpoint: /api/invite-friend

Request Headers:

- Content-Type: application/json
- Authorization: Bearer <access_token>

Request Body:

```
{
  "friend_email": "example@example.com",
  "message": "Hey, check out this cool app I'm using!"
}
```

b. START A BUG HUNT

HTTP GET End-point: /api/bughunt

This API retrieves a paginated response of all the current bug hunting programs available on the BLT platform.

Filter Hunts Options:

- Ongoing
- Incoming
- Ended

Request Body: None

Response Body (paginated):

```
{
  "programs": [
    {
      "id": "1",
      "name": "Bughunt1",
      "domain": "coolgame.com",
      "logo": "https://test.com/coolgame.png",
      "description": "Join our bughunt and win big prizes!",
      "prize_pool": 2000,
      "start_date": "2023-05-01T00:00:00Z",
      "end_date": "2023-05-31T23:59:59Z"
    },
    {
      "id": "2",
      "name": "Bughunt2",
      "domain": "coolgame.com",
      "logo": "https://test.com/coolgame.png",
      "description": "Join our bughunt and win big prizes!",
      "prize_pool": 1000,
      "start_date": "2023-05-01T00:00:00Z",
      "end_date": "2023-05-31T23:59:59Z"
    }
  ]
}
```

HTTP POST Endpoint: /api/bughunt

This API creates a new bug hunting program on the BLT platform.

Request Headers:

- Content-Type: application/json
- Authorization: Bearer <access_token>

Request Body:

```
{
  "id": "1",
  "name": "Bughunt1",
  "domain": "coolgame.com",
  "logo": "https://test.com/coolgame.png",
  "description": "Join our bughunt and win big prizes!",
  "prize_pool": 2000,
  "start_date": "2023-05-01T00:00:00Z",
  "end_date": "2023-05-31T23:59:59Z"
}
```

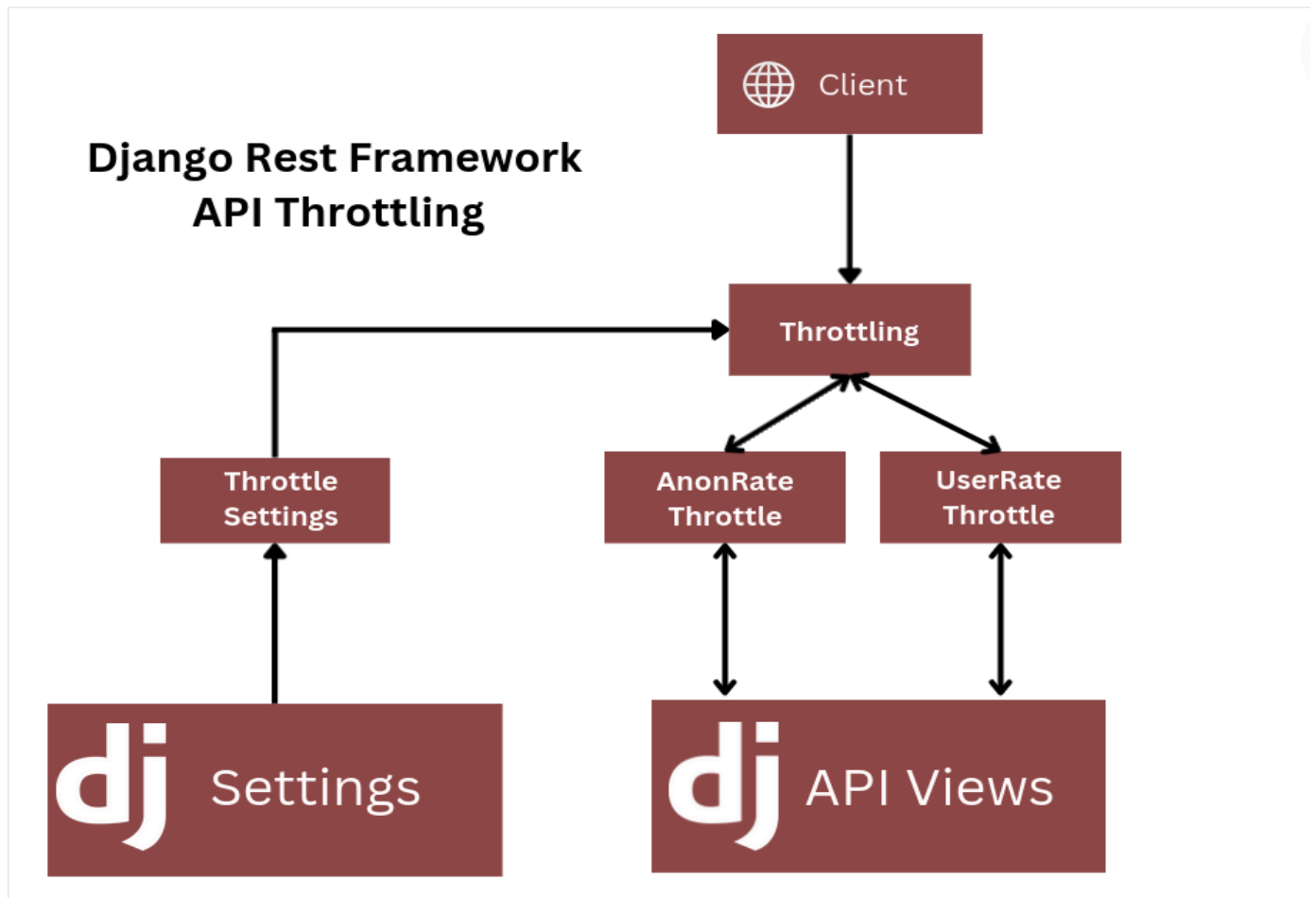
2. API RATE LIMITER

To protect our backend from abuse, integrating an API rate limiter is an essential part. This will help in limiting the API calls per user. Thus protecting the Backend from bot's and scripts trying to access BLT data.

Best tool which fits the case is Django-Rest-Frameworks [API Throttling](#)

Image and code given below gives a better understanding of how Django REST API Throttling works.

```
REST_FRAMEWORK = {
    'DEFAULT_THROTTLE_CLASSES': [
        'rest_framework.throttling.AnonRateThrottle',
        'rest_framework.throttling.UserRateThrottle'
    ],
    'DEFAULT_THROTTLE_RATES': {
        'anon': '2/min',
        'user': '4/min'
    }
}
```

BACON COIN

In order to encourage bug hunting and motivate participants, I propose to implement a reward system for bug hunters/reporters. To facilitate this, I plan to create an independent blockchain using Geth and the Proof of Authority (PoA) consensus algorithm. This approach is more energy-efficient than Proof of Work (PoW) and provides mining ability to only selected addresses and full nodes.

Our primary objective in implementing our own blockchain is to enable faster transactions and to have control over the network as the governing authority.



This will allow us to tailor the network to meet our specific needs and ensure its security and reliability.

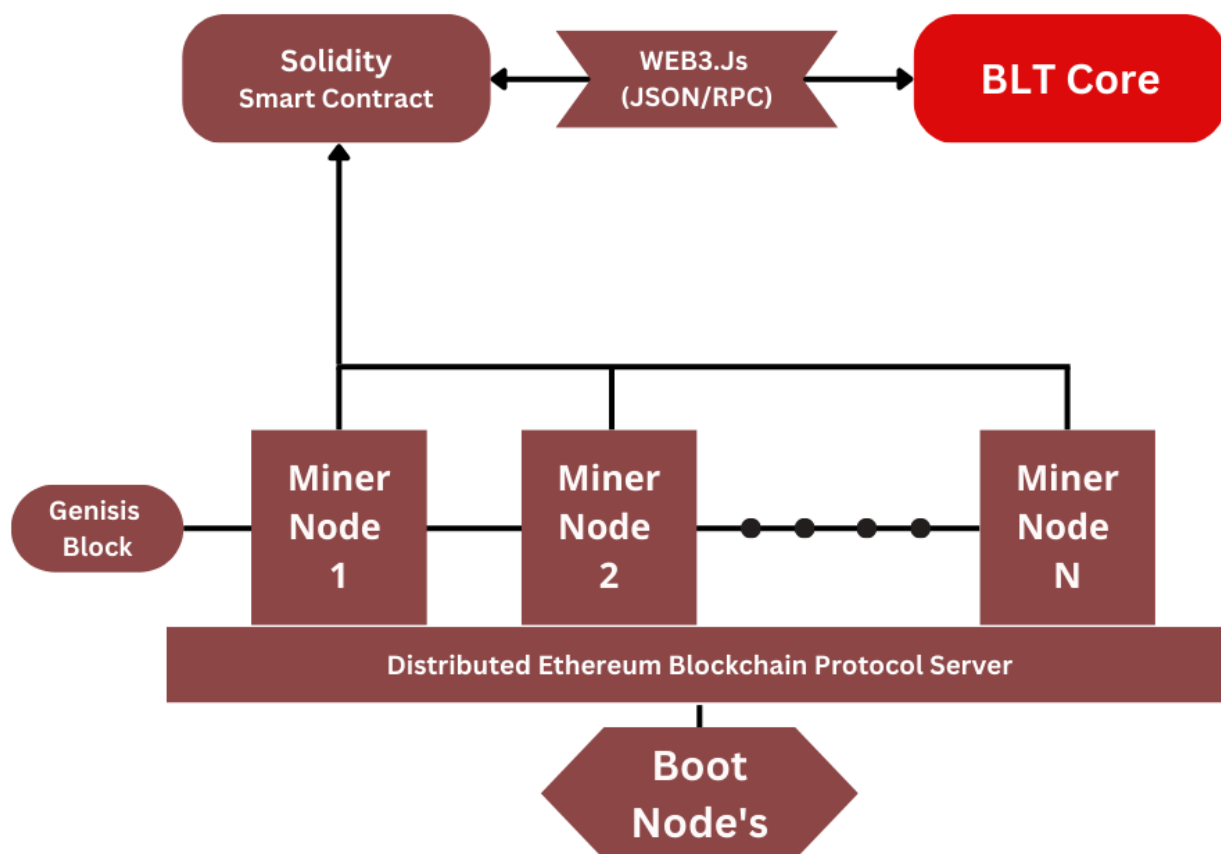
The rewards system will be implemented through a smart contract on the blockchain, which will distribute BACON Coins to bug hunters based on the severity of the bugs they discover. This will incentivize active participation and help to create a community of motivated bug hunters.

In addition to encouraging bug hunting, the BACON Coins can be used for other purposes in the future. For example, they can be exchanged for merchandise or discounts in our premium programs. We can also reward our active contributors with BACON Coins, which will further encourage participation and help to build a strong community.

SOLUTION AND WORKFLOW

1. Create a new blockchain using Geth and the Proof of Authority (PoA) consensus algorithm. PoA is a lightweight and energy-efficient alternative to Proof of Work (PoW) that still provides a high level of security.
2. Choose a set of trusted addresses and full nodes to serve as validators for the PoA network. These validators will be responsible for creating new blocks and verifying transactions.
3. Implement a smart contract on the blockchain that will serve as the reward system. This contract should be programmed to distribute BACON Coins to bug hunters/reporters based on the severity of the bugs they discover.
4. Launch the blockchain network and incentivize bug hunting by promoting it through various channels. Encourage your community to participate and offer attractive rewards to motivate them.
5. Collect bug reports through a dedicated channel (such as email) and verify their validity. Once a valid bug report is received, evaluate its severity and assign an appropriate reward based on a pre-determined scale.
6. Distribute BACON Coins to the bug hunter/reporter via the smart contract. This can be done automatically or manually, depending on your preference.

7. Monitor the blockchain network for any issues or bugs and perform necessary upgrades as needed. Keep the community informed of any changes or updates to the reward system.



ETHEREUM SMART CONTRACT

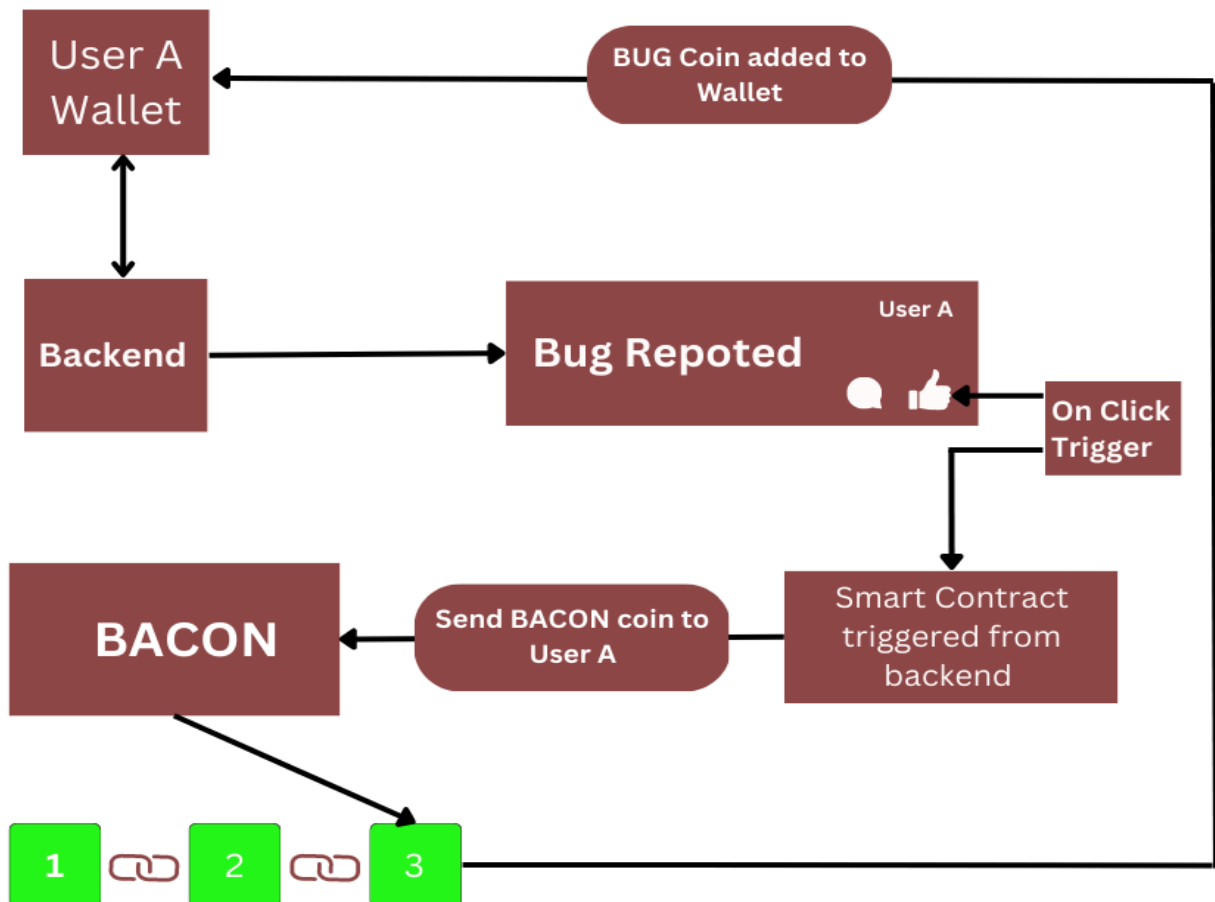
To Implement a full-fledged reward system we need to deploy a smart contract which is capable of handling all the distribution logic of the BACON coin over our BACON chain; this contract will be triggered from our BLT backend when a user meets the reward criteria.

I propose to create a smart contract using Solidity, which will facilitate the distribution of BACON Coins to bug hunters based on the severity of the bugs they discover. The smart contract will also enable the exchange of BACON Coins for merchandise and discounts in our premium programs. I will deploy this smart contract on our independent blockchain using Geth and the Proof of Authority (PoA) consensus algorithm.

- To implement the BACON Coins reward system, a Solidity smart contract needs to be developed with clear requirements in terms of rules for distributing the coins to bug hunters based on the severity of bugs reported, the total number of coins in circulation, and the types of merchandise and discounts available for redemption.
- Once the Solidity code has been developed, it must be tested locally using a development environment such as Remix to ensure that it is functioning as intended. The code is then compiled to create a bytecode representation of the smart contract.
- To deploy the smart contract to the independent blockchain, a tool like Remix or Truffle is used. The deployment process involves specifying the address of the governing authority and other configuration settings. The smart contract code is verified to ensure that it has been deployed correctly and is functioning as intended.
- The smart contract is then tested on the blockchain using simulated transactions to ensure that the distribution of BACON Coins and the exchange of coins for merchandise and discounts are working as expected. Ongoing monitoring of the blockchain and the smart contract is essential to ensure that all transactions are being processed

correctly and that there are no errors or vulnerabilities in the code.

- Any necessary updates to the smart contract code are made to fix any bugs or to make improvements to the rules for distributing BACON Coins or exchanging them for merchandise or discounts. The Solidity code is refactored and optimized to reduce gas costs and improve performance.
- Documentation of the smart contract and the deployment process is essential to enable others to use and understand the code, including any configuration settings and dependencies. By following these steps, a functional and secure smart contract can be developed to facilitate the BACON Coins reward system.



POSSIBLE SMART CONTRACT FUNCTIONS

```
pragma solidity ^0.8.0;

contract BugBounty {
    uint public totalBaconCoins;
    address public owner;

    constructor() {
        totalBaconCoins = 10000; // Total number of BACON Coins in circulation
        owner = msg.sender; // Set the contract owner
    }

    // Define a struct to represent a bug report
    struct BugReport {
        string title;
        string description;
        uint8 severity;
        bool isFixed;
    }

    // Define an array to store the bug reports
    BugReport[] public bugReports;

    // Define a mapping to keep track of how many BACON Coins each address has earned
    mapping(address => uint) public baconCoinBalances;

    // Define a function for submitting a bug report
    function submitBugReport() public {

    }

    // Define a function for calculating the number of BACON Coins to award for a given bug
    severity
    function calculateBaconCoins(uint8 _severity) public pure returns (uint) {

    }

    // Define a function for awarding BACON Coins to a bug hunter
    function awardBaconCoins(uint _index, address _hunter) public {

    }

    // Define a function for redeeming BACON Coins for merchandise or discounts
    function redeemBaconCoins(uint _amount) public {

    }
}
```

PRIVATE ISSUE REPORTING

In some cases, bugs can be highly critical and should not be disclosed publicly. To address this, companies should have the option to create a private bug hunting program, in which all bugs reported by participants are kept private by default. The company administering the program would be the only entity with access to these vulnerabilities.

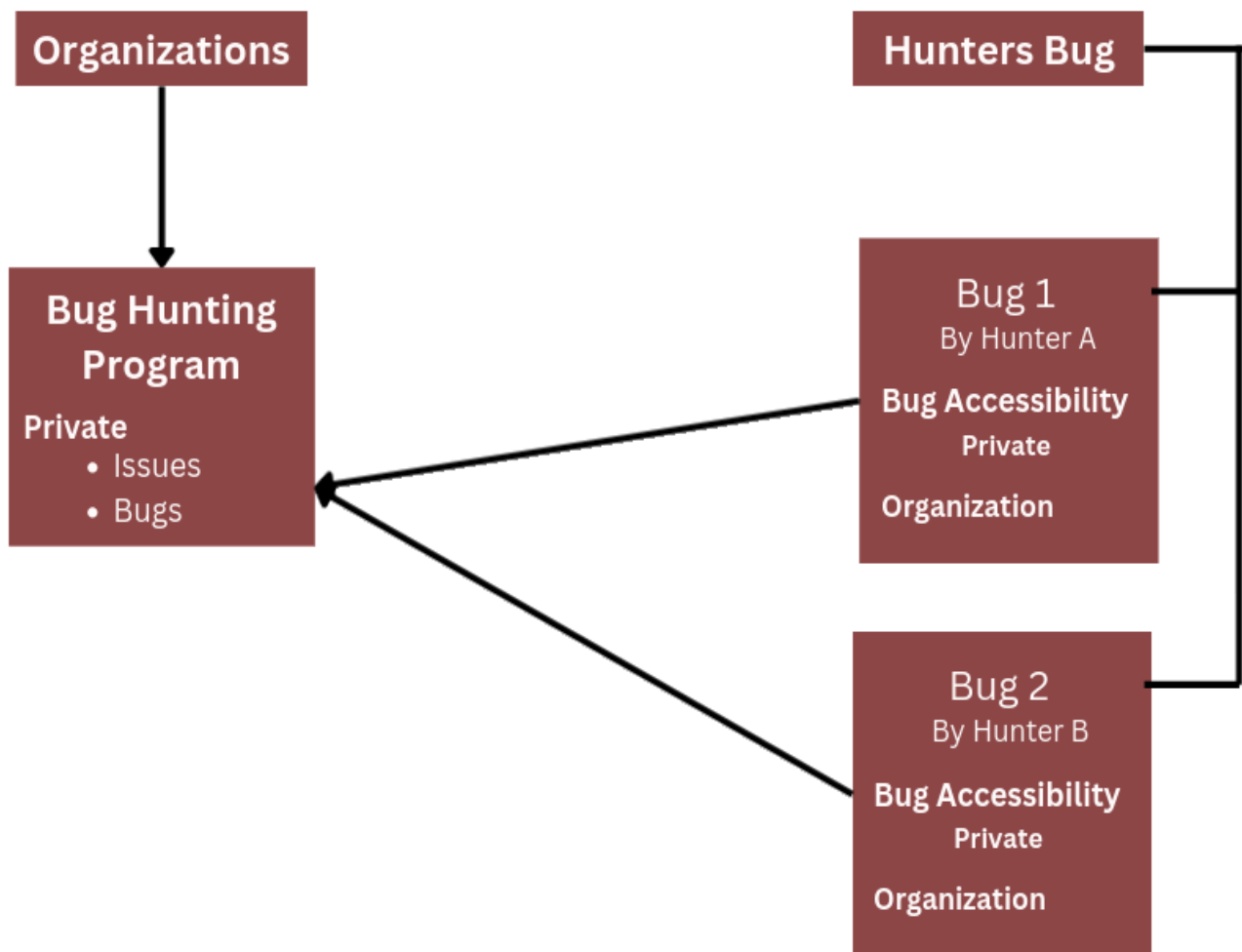
If a company marks them as privately reported organization then all the bug reported to their domains will automatically be reported as Private Issues to the company

Additionally, users should have the ability to tag specific organizations or companies to keep the bug/issue private and accessible only to the tagged organization. This would ensure that sensitive information is only shared with those who have a legitimate need to know.

WORKFLOW

1. To address the issue of highly critical bugs that cannot be disclosed publicly, companies should have the option to create a private bug hunting program.
2. The company administering the program sets up the program and invites trusted individuals or third-party companies to participate.
3. Participants can report bugs they discover in the program, with the understanding that all bugs will be kept private by default. Participants may also tag specific organizations or companies to keep the bug/issue private and accessible only to the tagged organization.
4. The company administering the program verifies the reported bugs and determines the appropriate response. The response may include patching the bug, compensating the participant for their contribution, further investigation, or commenting on the Bug.

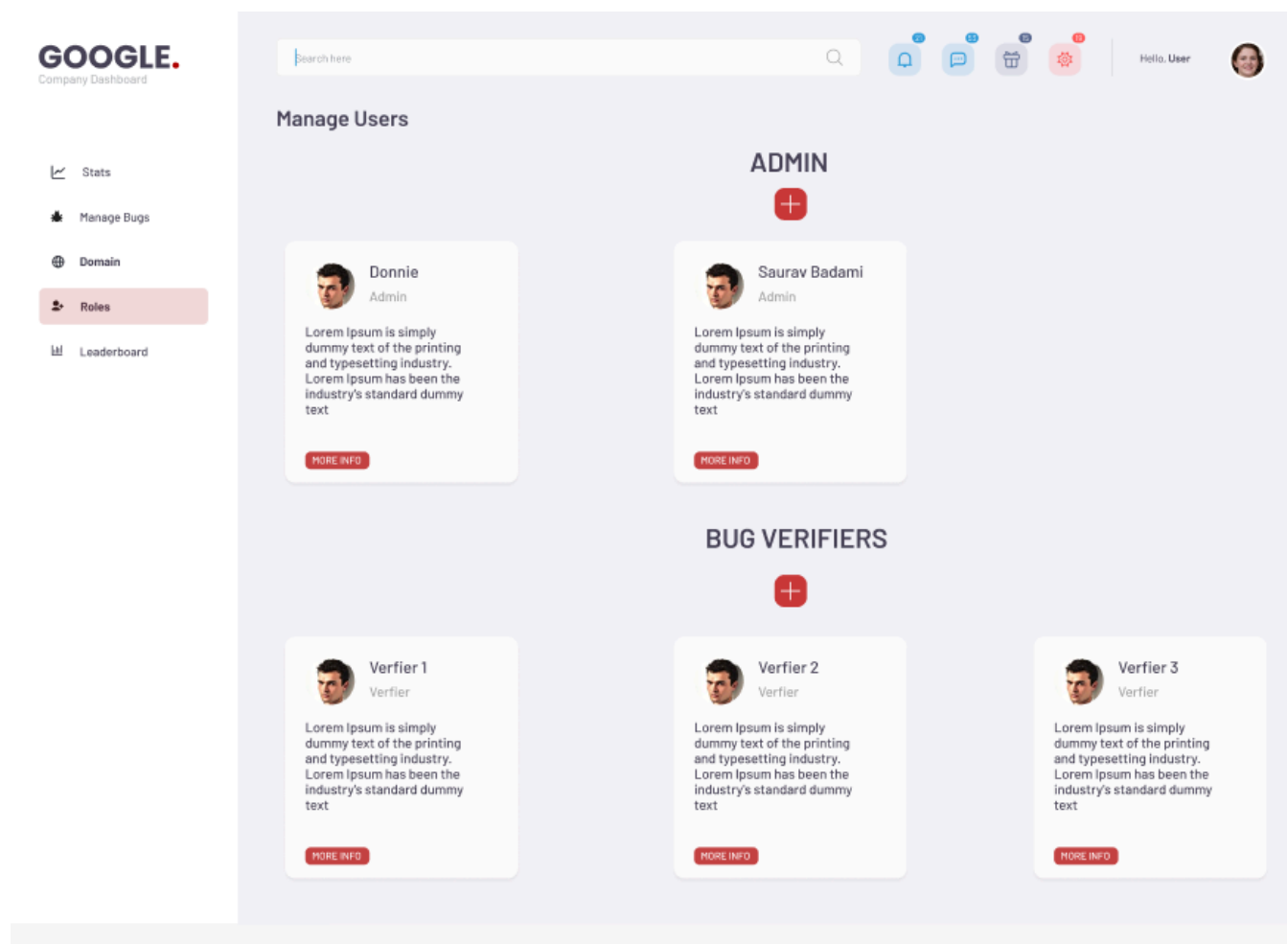
5. Once the bug has been fixed, the company determines whether or not to publicly disclose the bug, depending on the severity of the vulnerability and potential impact on customers.
6. The company should follow up with participants to ensure that they are satisfied with the response and compensation, if applicable. Additionally, the company should evaluate the effectiveness of the private bug hunting program and make any necessary improvements.



COMPANY DASHBOARD

User Management

The first section of the dashboard would be User Management, where the company creator or admin can add multiple users to the company and assign them different roles such as **admin, manager, bug verifier, etc.** This section will allow the admin to manage the users and their roles within the company.



Domain Management

The Domain Management section will allow the company to create multiple domains and start bug hunts on those domains. The company can create, edit and delete domains and assign bug verifiers to each domain.

Company Stats

The Company Stats section will provide statistical data about the bug hunts performed on different domains. The stats will include:

- The number of Open bugs, Closed bugs, and Total bugs for each domain.
- Stats for different types of bugs reported on each domain.
- Stats for the number of bugs reported per day in a barchart.
- Money spent on each bug hunt.



Manage Bugs

The Manage Bugs section will allow the Bug verifiers to see all the Bugs reported for their assigned domains. Initially, the bugs will be private, and the Bug verifier will verify if the bug is genuine or not. If the Bug verifier finds it genuine, then they can mark it as verified, and the user who reported the bug will be awarded BUG COINS. After the Bug is resolved, the company can choose to make it Public, allowing the bug to be seen by other users for educational purposes.

Top Testers Leaderboard

The Top testers Leaderboard section will display the top testers for that company, filtered by domains. The leaderboard will show the users who reported the most number of bugs, and the company can offer rewards to those users.

In conclusion, the proposed company dashboard will allow the company to manage its bug hunts, domains, and users efficiently. The dashboard will provide valuable insights into the bug hunts performed on different domains and allow the company to reward its top testers.

I am planning to implement Manage Bugs, Top Testers leaderboard, User management in Company dashboard.. If time permits, I may also work on additional features.

PROPOSED TIMELINE

Pre GSoC Period	
Before May 4	• Study about BLT core. • Get a good understanding of the problem statements. • Research about Ethereum infrastructure in depth and POA consensus. • Ask questions.
Community Bonding Period	
May 4 - May 28	• Interact with mentors and community members. • Review/revise the Proposal design and deliverables. • Create New figma designs for Start Bug Hunt, Report bug and Company dashboard.
Coding Period	
May 29 - June 5	• Implement Start Bug Hunt and Report a Bug page. Integrate them with BLT backend
June 5 - June 15	• Implement User Management, Manage Bugs and Top testers leaderboard for Company dash board.
June 15 - July 1	• Implementing Private Issue reporting .
July 2 - July 6	• Start working on API rate Throttling/Limiting.
July 6 - July 14	• Complete the leftover work for 1'st Phase of Evaluation.
Mid-Term Evaluation	
July 14 - July 16	• Start learning more about Geth and running nodes on 0 gas price. • Learn about deploying ethereum smart contracts in independent chains.
July 17 - July 25	• Start creating the independent chain using geth and deploy 4 mining nodes + 1 bootnode.
July 26 - Aug 7	• Create and deploy the smart contract on Bacon.

Aug 7 - Aug 15	Integrate Smart contract with backend using web3.js
Aug 15 - Aug 21	Finish some leftover work, document and write tests for the API's.
Final Evaluation	

DELIVERABLES

Mid-Term Evaluation

- New Figma designs for Start Bughunt, List Bughunt and Report A Bug page (<https://github.com/OWASP/BLT/issues/1228>)
- Report A Bug Page in tailwind (<https://github.com/OWASP/BLT/issues/1231>)
- Feature Private Issue Reporting (<https://github.com/OWASP/BLT/issues/1237>)
- API rate Throttling/Limiter (<https://github.com/OWASP/BLT/issues/1234>)
- BACON COIN Over Independent chain on Metamask (<https://github.com/OWASP/BLT/issues/1235>)

Final Evaluation

- Deployed Independent chain with 4 mining nodes + 1 bootnode (<https://github.com/OWASP/BLT/issues/1235>)
- Deployed Smart contract over BACON Chain (<https://github.com/OWASP/BLT/issues/1236>)
- Reward system integrated with backend (<https://github.com/OWASP/BLT/issues/1236>)
- Bug Hunt
 - Start Bughunt Page + Private Bughunt (<https://github.com/OWASP/BLT/issues/1229>)
 - List Bughunt Page (<https://github.com/OWASP/BLT/issues/1230>)
- Company Dashboard (<https://github.com/OWASP/BLT/issues/1238>)
 - Top Leaderboard
 - Manage Bugs
 - User Management
- Home page



-

ALTERNATIVE DELIVERABLES (Final Eval)

- **Implement New Homepage**
- New Detail Issue page which will show all new features we added in the report bug page

COMMITMENTS

My university will not be providing any sort of vacation due to disturbance in academic timeline because of the Corona Pandemic, but the attendance is not mandatory thus I will be able to provide 40 hrs per week. I might have exams between June-July, but I'm targeting a minimum commitment of 20hrs per week.

The proposed duration of the project is 350 hrs.

WHY ME?

With a year of active contribution at BLT and interacting with the mentors. I have developed a deep understanding of the project's codebase and identified potential areas for optimization. My 2-year experience with backend and deployment technologies, combined with my participation in hackathons involving blockchain development and my core understanding of blockchain concepts which I've gained from participating in Summer of Bitcoin, has equipped me with the necessary skills to contribute significantly to the project.

When I got to know that BLT is planning to integrate Blockchain in their ecosystem I was really excited and started working immediately, thinking of possible solutions to make it possible.

Even if this project wouldn't be selected for GSOC, I still would work on this



project during my summer vacations and is ready to risk my participation in GSOC by submitting a proposal to only 1 project.

Moreover, my proficiency in using Chat GPT adds to my technical expertise, allowing me to make valuable contributions to BLT. I am confident in my ability to leverage these skills to drive progress and success in the project.

AFTER GSOC

Upon completing my GSOC, I am confident that I will have gained an in-depth understanding of the OWASP BLT codebase, enabling me to confidently make substantial improvements to the project.

I am committed to continuing my contributions to OWASP BLT, as well as other OWASP projects, by implementing innovative ideas that align with the organization's objectives.

If possible I will also try to maintain BLT and increase community engagement of BLT by posting about BLT on socials.

REFERENCES

- https://owasp.org/www-community/initiatives/gsoc/gsoc_sat
- <https://owasp.org/www-community/initiatives/gsoc/gsoc2023ideas>
- <https://github.com/OWASP/BLT-Flutter>
- <https://github.com/owasp/blt/>
- [Figma Designs](#)