GSoC'23: OWASP

Bug Logging Tool (BLT)

Mentors:

Donnie

Aryan Ranjan

Personal Details

Name: Amrit Prakash

GitHub handle solo-daemon

LinkedIn handle amrit-prakash

Phone no. +91 9693832520

Email amritdevil1st@gmail.com

Location 🇮🇳India

Time-Zone India (UTC +5:30)

University Indian Institute of Technology Roorkee

Degree Bachelor of Technology (4-year Program)

Current 2nd Year (Batch 2025)

About Me

My name is **Amrit Prakash,** currently pursuing B.Tech at **the Indian Institute of Technology Roorkee** with a major in Mechanical Engineering. I am a full stack developer, currently **Project Associate** at Information Management Group, IIT Roorkee. I like development and new technologies excite me. In my initial college days, I focused on web development and mobile development. I also like to explore Web3 technology. I have also created a project on **blockchain technology Webster**[Github Repo] for Syntax Error, 36-hour long hackathon. My Team won **10 NFTs** from one of its sponsors in this hackathon. So you can say that I like to explore every field of technology.

Coding Skills:

I would like to summarize my coding skills below:

- Fluent in Flutter SDK & the Dart programming language.
- Fluent in C++ with Sound knowledge of OOPs.
- Proficient in Python (including OOPs).
- Fluent knowledge of Django and Django Rest framework.
- Introductory experience with BlockChain Development in Remix and Solidity.

Other Skills

- Experienced in UI design with Figma.

Development Environment

- Windows 11 for Flutter development and WSL for the Django backend.
- VS Code & Android Studio.

Statement of motivation

What is your motivation for participating in Google Summer of Code?

I love the thought of people using things I have built. Even the slightest feature and detailing excite me. These are the things that brought me into software development. By participating in GSoC, I get exposure to how software development happens on the industrial level. It also allows me to get acquainted with the Open Source World where high-quality coding standards are achieved in a fun and interactive way. Lastly, it's an excellent opportunity to learn from experienced mentors, something that I don't want to miss!

Why did you choose OWASP, and why this project idea?

Information security has become a major issue in the modern world so much so that there are roles like ethical hacking, Identification of bugs is an important step in making an application better and safer. OWASP is an org that is working towards making the internet a safer place and I want to be a part of this journey by working with them.

Identification of bugs is an important step in the process of making applications safer and by working on this project I want to contribute to the very fundamentals of Information security and testing i.e., bug reporting. I chose this project because I am well acquainted with the codebase and fluent with the tech stack required. I already have worked on projects in different domains but I still feel the requirement of brushing my Flutter skills so this project also gives me a learning opportunity to increase my knowledge of the above-mentioned frameworks and languages.

Contributions to OWASP

I have 19 PRs merged in the **BLT-FLUTTER** repository and 1 PR merged in the **BLT** repository. I have opened 19 issues in the BLT-FLUTTER and am always open to discussion. I have gained a keen interest in the project and have always tried to explain my PRs and what problems they solve.

Merged Pull Requests:

BLT-FLUTTER

- Display user score on profile page
- Fixed upvoted issues section on Profile page for anonymous user
- Change Recent Activity Section on Profile Page
- State Change on Start a Bug Hunt
- Search Bar close on double tapping clear icon
- Set orientation Lock on the App
- Open Urls in Company Detail Page .
- Paste Image From Clipboard
- Company Detail Page Site now opens in a browser
- Build Error due to CompileSdkVersion
- Stack Optimization and Null Check Error Fix
- Prevent abrupt logout from Home Page
- Fix Onboarding screen appear multiple times in the lifecycle of app
- Feature to Display Monthly LeaderBoards
- Show Different Months Data in MonthlyLeaderBoard
- Feature Social Page
- Social Page
- Scroll Fix on Terms of Service Page
- Add category/labelfield to issue form

BLT:

- Generate Data for Monthly LeaderBoard

Opened Issues:

BLT-FLUTTER:

- The upvoted issue section for anonymous user fails
- Upon clicking on Explore anonymously on welcome page null value check error occurs
- Start Bug Hunt Button does not changes state of Report Bug Page
- Search bar should close if the clear icon is pressed and query is empty
- Feature for opening urls from the Company Detail Page
- The site button on Company Detail Page doesn't opens the url in an external browser .
- On running the "flutter run" it command gives an error to bump up the compile sdk version
- The current stack management is unoptimized
- Prevent abrupt logout from home page on pressing back
- Monthly Leaderboard displays global leaderboard results.
- Onboarding Screen opens up when we open the app in logout state .
- Api not working due to wrong baseUrl
- Show every month's leaderboard of a year
- Feature : Social Page
- Show multiple social accounts of blt in social pages like github , facebook ,twitter etc.
- Scroll problem on terms of service page
- Category/label Field to issue form

BLT:

- Monthly Leaderboard is showing global leaderboard results

Other Open-Source Contributions

- IMGIITRoorkee
- Rocket.Chat

Development Experience:

Some of the project which I maintain or are made by me are listed as follow.

- **Pursuit**[GithubRepo]**:** A **Django** and **React** project that assists campus clubs with their recruitment process. This is also the project where I have used **DRF** which is required in my project.

- **Omniport Docker** [Github Repo]: The Dockerized setup of the one true portal for any and every educational institute.

- **Buy-and-Sell** [Buy-And-Sell Frontend**,** Buy-And-Sell-Backend]**:** An app for people to buy and sell goods with other people on the IIT-R Campus.

Project Description

A brief overview of the project

I propose a duration of **350 hrs** for this project. The idea involves improving the current **UI-UX** as well as adding **new features** to the app along with the corresponding work required on the **Django API.** Here are some brief points for why my project is required:

- **API and Tests:** I am proposing the corresponding API and Tests required for the below-mentioned features.

- **Filters:** There are no filters used anywhere on the app except for in the search and leaderboard. I propose introducing filters. e.g. On the **Issue page,** there can be filters based on **user, label, and status,** and similarly for **Bug-Hunts**. The **Your-Issues** or **Your Bug-Hunt** is proposed as a special filter that will also provide edit access for these issues and bug-hunts. A detailed description of all filters is in the **Project's Major Flows** Section

- **Search Filters:** The same filters discussed above can be used for search filters.

- **Editing Issue and Closing Issue: [Feature]** There is no way a user can **edit** or **close** their **reported issues**. I propose adding a feature with which a user can edit the issues reported by them. Same for **closing** an Issue but closing privilege can also be given to other users as explained below. However, if the user closes the issue themselves the issue will be marked **unverified.** A detailed logical description is later in the proposal.

- **Comment: [Feature]** User can comment on issues on the Issue Detail Page. It will also include a feature to reply to a comment of the issue.

- **Modify Report-Bug Form**: We are currently using the **url field** to create a **issue**. Since we have a **domain field** as a foreign key of the issue. I propose adding that to the form. Then we can easily track which bug belong to which company.

- **Anonymous Bug Reporting: [Feature]** An anonymous user or even a normal user will be able to report bugs anonymously.

- **Draf Hunts and Issues:** User can create draft hunts and issues which will not be visible to other users.

- **Bug-Hunt: [Feature]** Currently, the user cannot see any Bug-Hunt either active or previous. In my proposal, I have included designs and flows on how this can be achieved. The **start Bug Hunt** feature is also a part of it, but it depends on the availability of the **payment method**.

- **Bug-Hunt Dashboard Page:[Feature]** User wil be able to view all the info about a bug hunt it's **issue**, **results** and **leaderboard**.

- **Company Detail Page: [Feature]** Users can view info about a **company,** it's **top hunters, bug hunts, and issues linked to the company.** This page will include the section **Issue, Bug-Hunt, Activity, and LeaderBoard** for the company**.** A detailed discussion is later in the proposal.

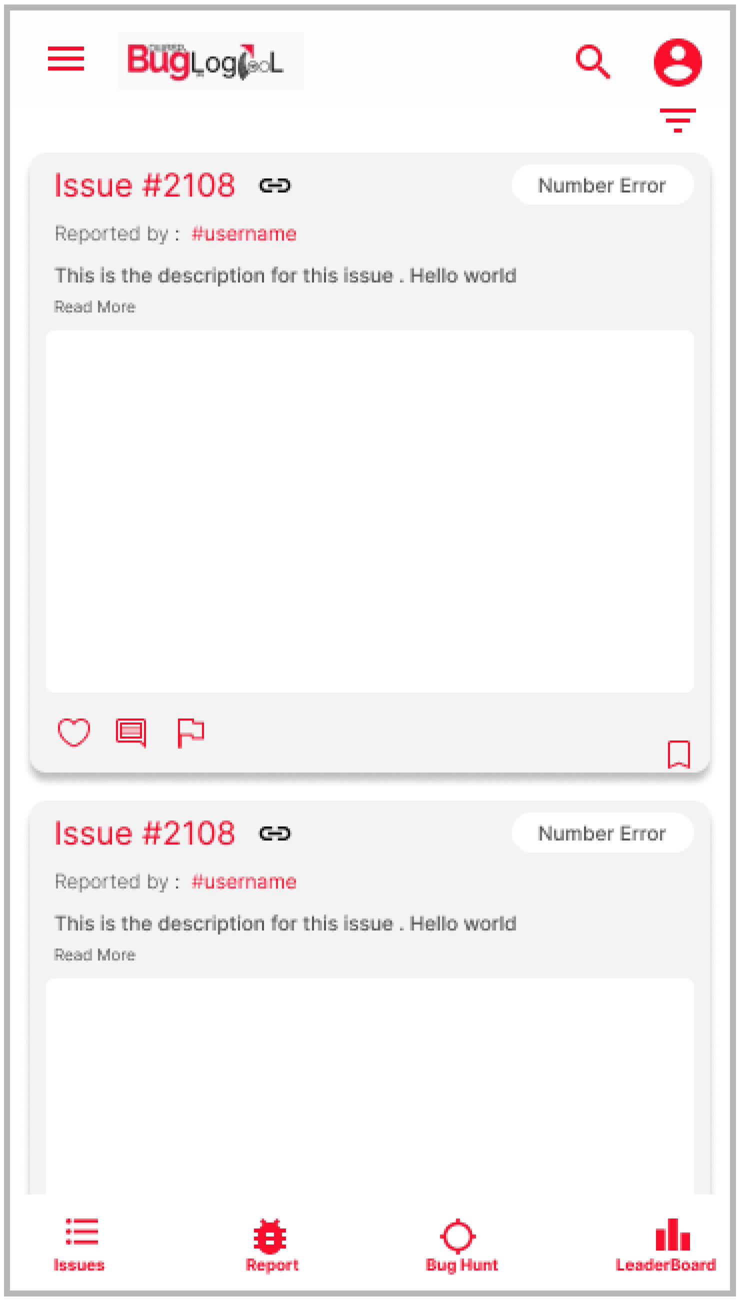
  Note: Generating data for Company Activity is not a part of the project. However, a simplistic approach can be implemented.

- **Company Dashboard Page: [Feature]** The admin users of a company will be able to view all the data on the **Company Page,** as well will have the privilege to **Close Issues(as verified), and Release Bug Hunt Results** on behalf of the company. Also, the admin users will be able to assign **roles.**

UI and UX: Reforms or Additions:

All the **Figma** screens used below can be found here.

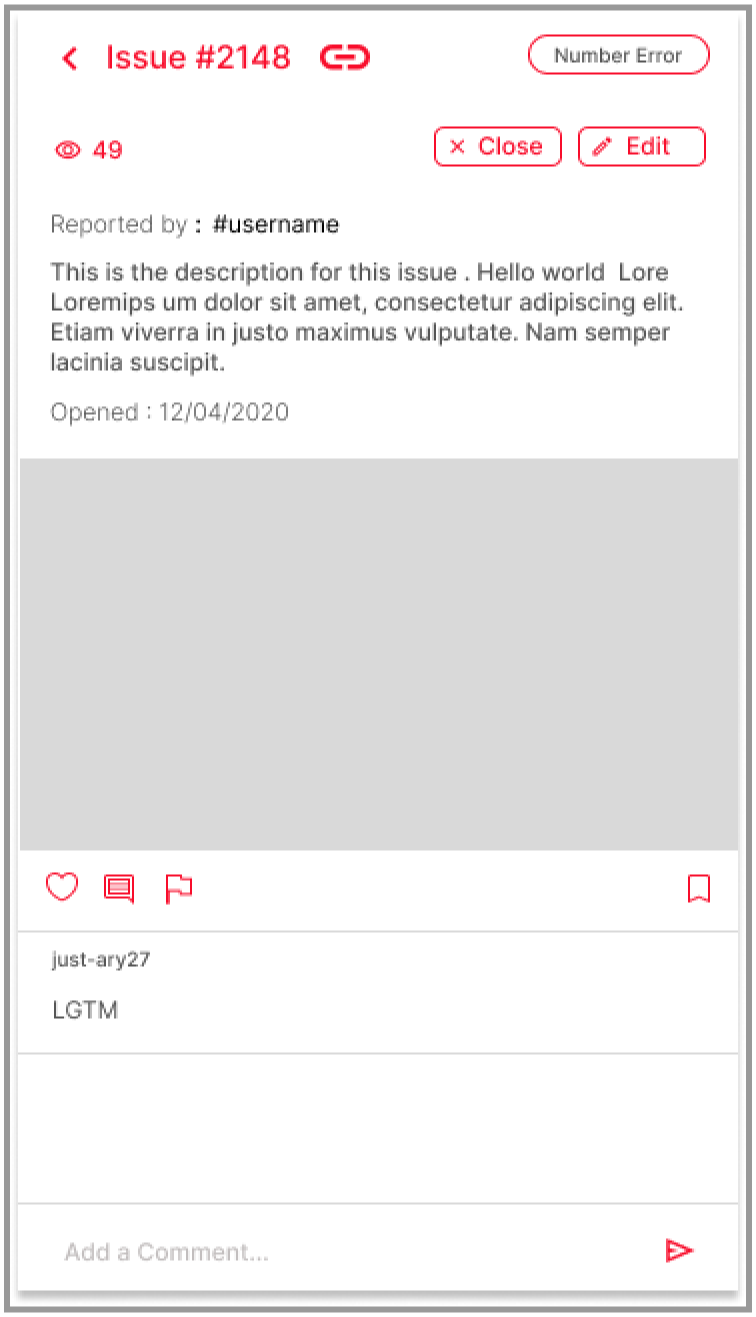
Issue Page: [Reform]



- **Filters** will be used for navigating between different types of issue filters including filters on the **status** of the issue, the **type** of the issue, issue **labels,** and **your issues** section.

- **New-Issue-Card** will help us in presenting the necessary data such as **issue-URL, issue-label, issue reporter, issue-description,** and **images or screenshots associated** with the issue and will also allow the user to perform their actions from this page which are **like, comment, flag,** and **save.**
- **Your-Issues** sections will contain issues that were reported by the user and will contain an option for **editing active issues.**

Issue Detail Page: [Reform]



Reported by : #username

This is the description for this issue . Hello world Lore Loremips um dolor sit amet, consectetur adipiscing elit. Etiam viverra in justo maximus vulputate. Nam semper lacinia suscipit.

Opened : 12/04/2020

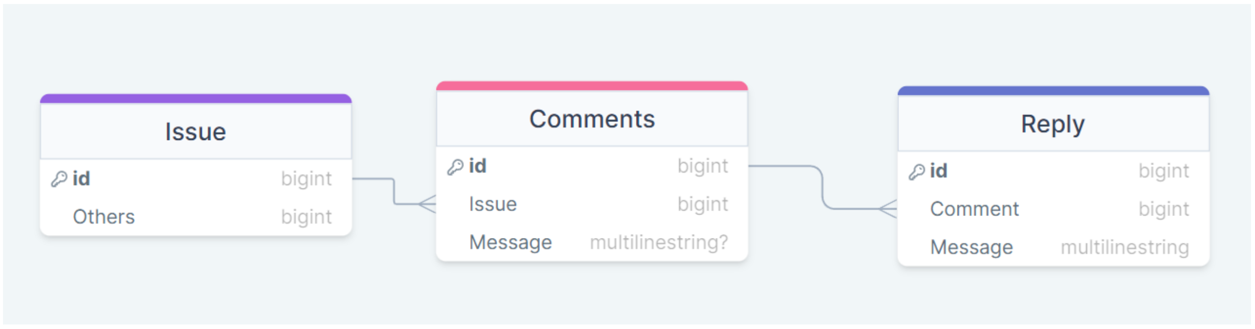
just-ary27

LGTM

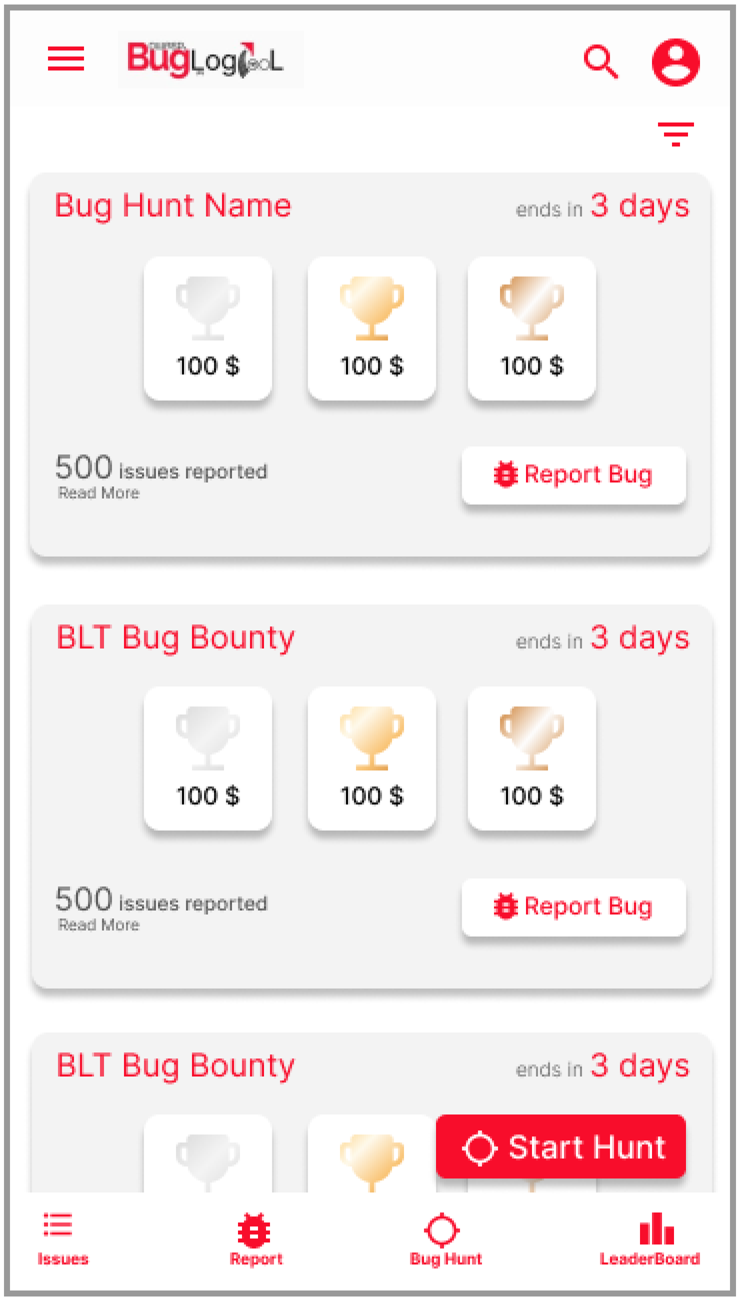
- **Comment:** Users will be able to comment on the issue
- **Detailed-Description of the issue:** This will try to serve all the useful data to the user which is all data and action of **issue-card** and additional data which involve **maximum** utilization of the current redundant fields from **API.**
- **Edit Access:** Only the user who has opened the issue will be able to edit the issue.
- **Close Access:** The detailed description of this control is later in the proposal.

The screen shown is for a **user** who **reported** the **issue.**

Implementation of **Comments** and **Reply**: The proposed solution for the comment is to make a **comment model** and **reply model** the comment model will have an issue object that has a foreign key and the reply model will have a comment object as a foreign key.


Issue
id bigint
Others bigint
Comments
id bigint
Issue bigint
Message multilinestring?
Reply
id bigint
Comment bigint
Message multilinestring

Bug-Hunt Page : [Addition]



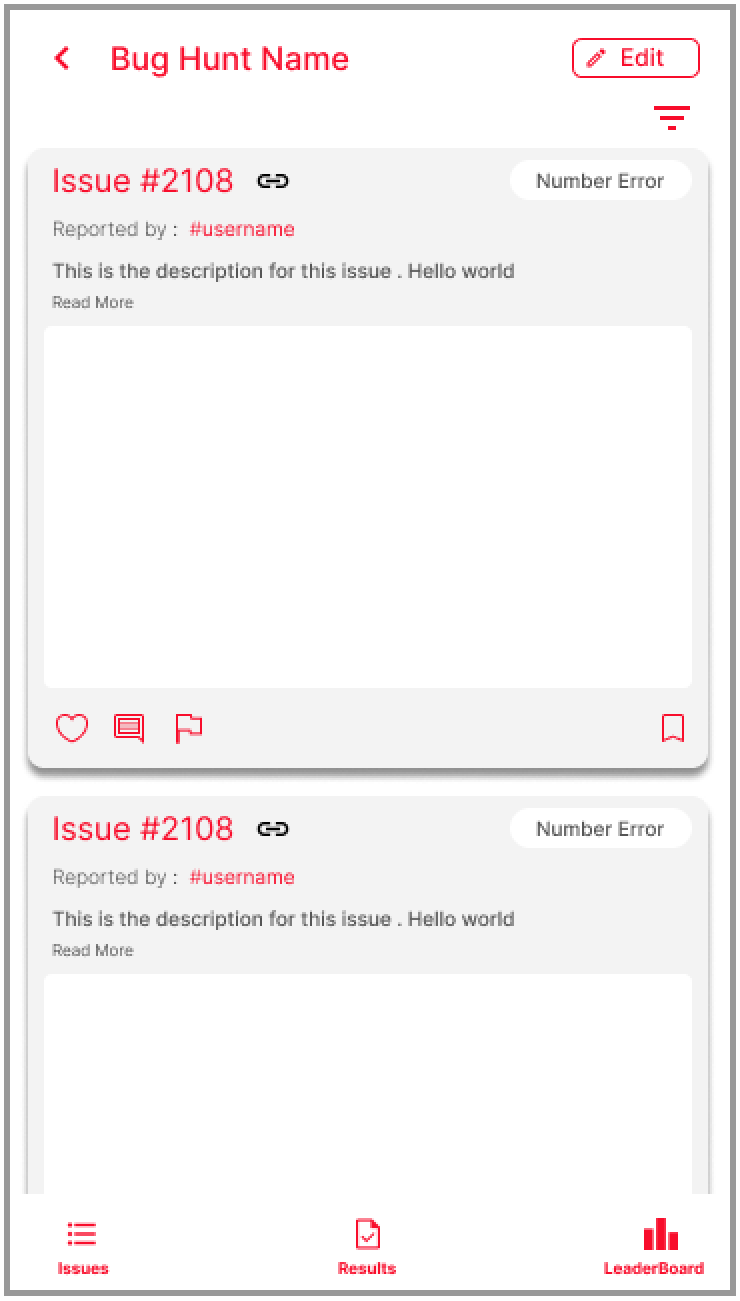
- **Filters: Filters** will be used for navigating between different types of bug-hunt filters, based on their status.

- **Active, Past, and Upcoming,** of the issue, the **type** of the issue, and **your issues** section.

Bug-Hunt-Card: Will display important data like **prizes, the number of Issues reported, the domain** the bug hunt was started for, and the **company** which started it

Start-Bug-Hunt-Button: Obviously one needs to start a **Bug Hunt.**

Bug-Hunt-Info-Page : [Addition]



- **Filters**: For **issue page** on this screen, is the same as the issue page.

- **Results:** The user will be able to see the results of the bug Hunt on the **bug-hunt-result** page.

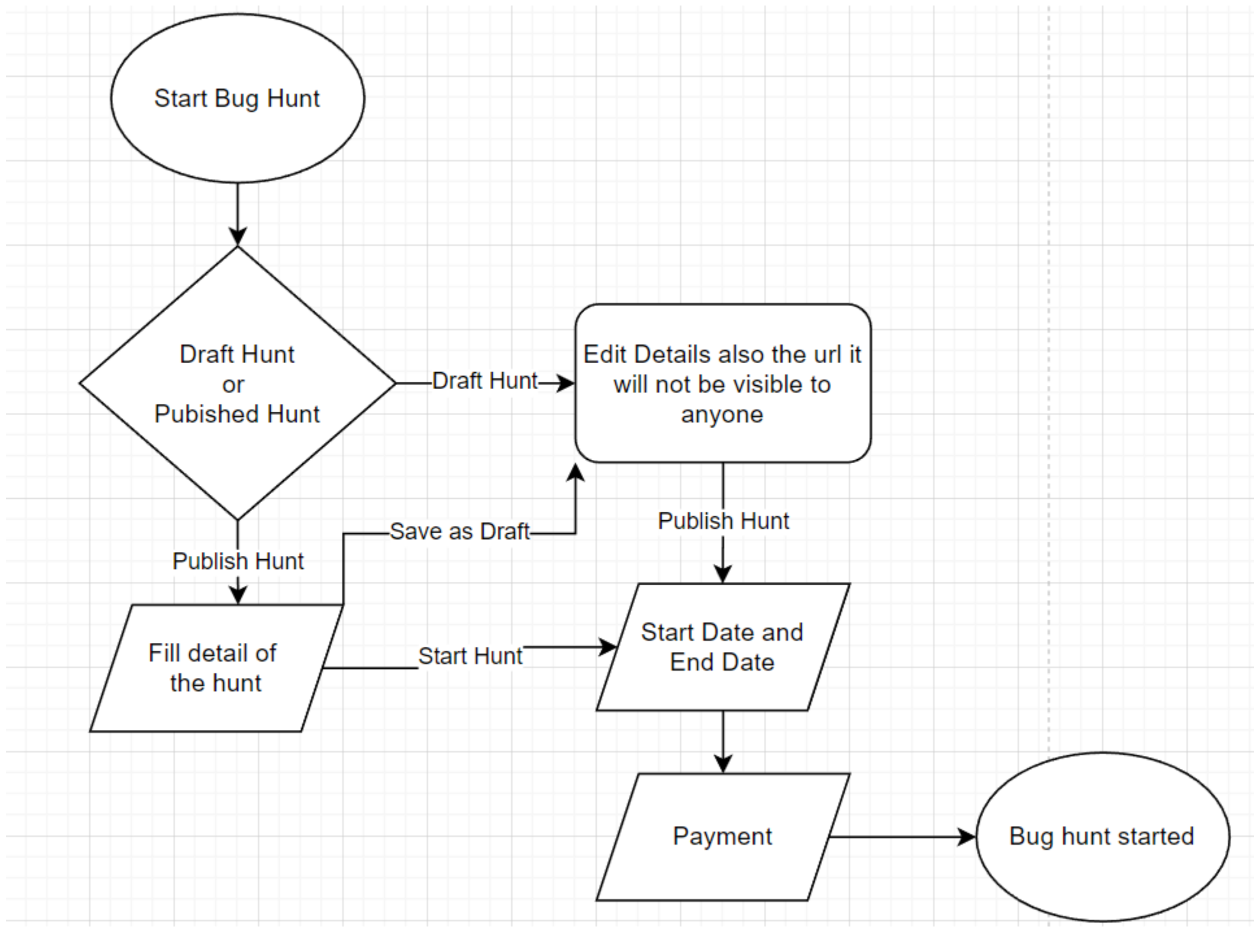
- **Leaderboard**: The leaderboard for the bug hunt will be generated here. This will use the same **algorithm** we use for the **global leaderboard.**

- **Edit:** The user or the company which created the bughunt can edit the bug hunt's start date, end date etc.

This screen shows all the issues reported for this bug hunt.

Bug Hunt-Dashboard/Start-Bug-Hunt-Page : [Addition] For the user who started the bug-hunt. Can **edit** a **draft hunt** or publish a new one

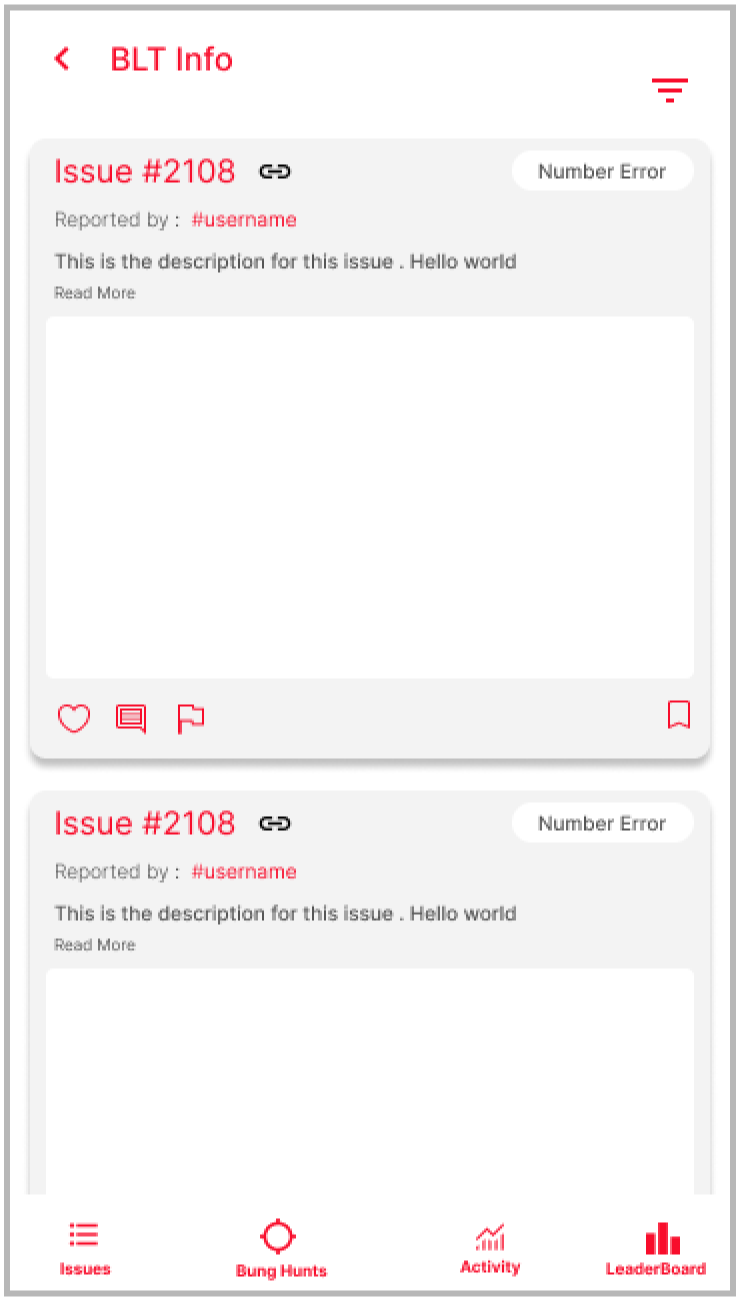
- Basically, a **form** that will be used for creating a bug hunt.
- Same style of UI could be used for the draft-hunt or manage-hunt page so the user can just edit and update his issues



UX Flow for starting a **bug hunt**

Search Page: [Reform]: The user will be able to search with filters. Also, users can search for other **users** and **bug hunts**

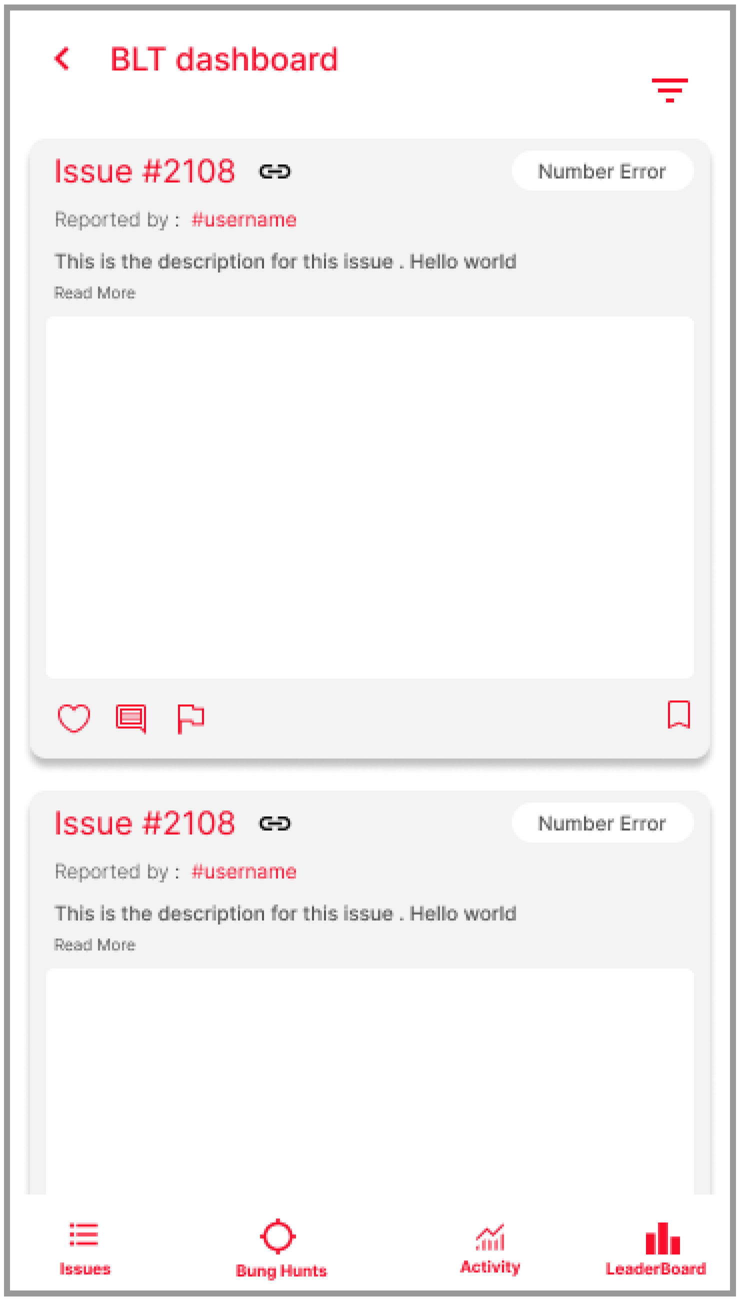
Comapny-Detail-Page :[Addition] This will be an extension of the company detail page i.e., we will use a **SliverAppBar** to show data related to the existing **company detail** page in the the page **app-bar**. Upon scrolling the screen below will show the **company-info** page



- **Issue Page:** Similar to the normal issue page but only for the issues associated with this company's domain.
- **Bug Hunt:** Similar to a normal bug hunt page but only for the bug hunts associated with this company's domain.
- **LeaderBoard:** Similar to the current **global leaderboard** but for the company.
- **Activity:** A page where all the activity for the company. However, it is not included in my **proposal**.

Company-Dashboard-Page:[Addition]

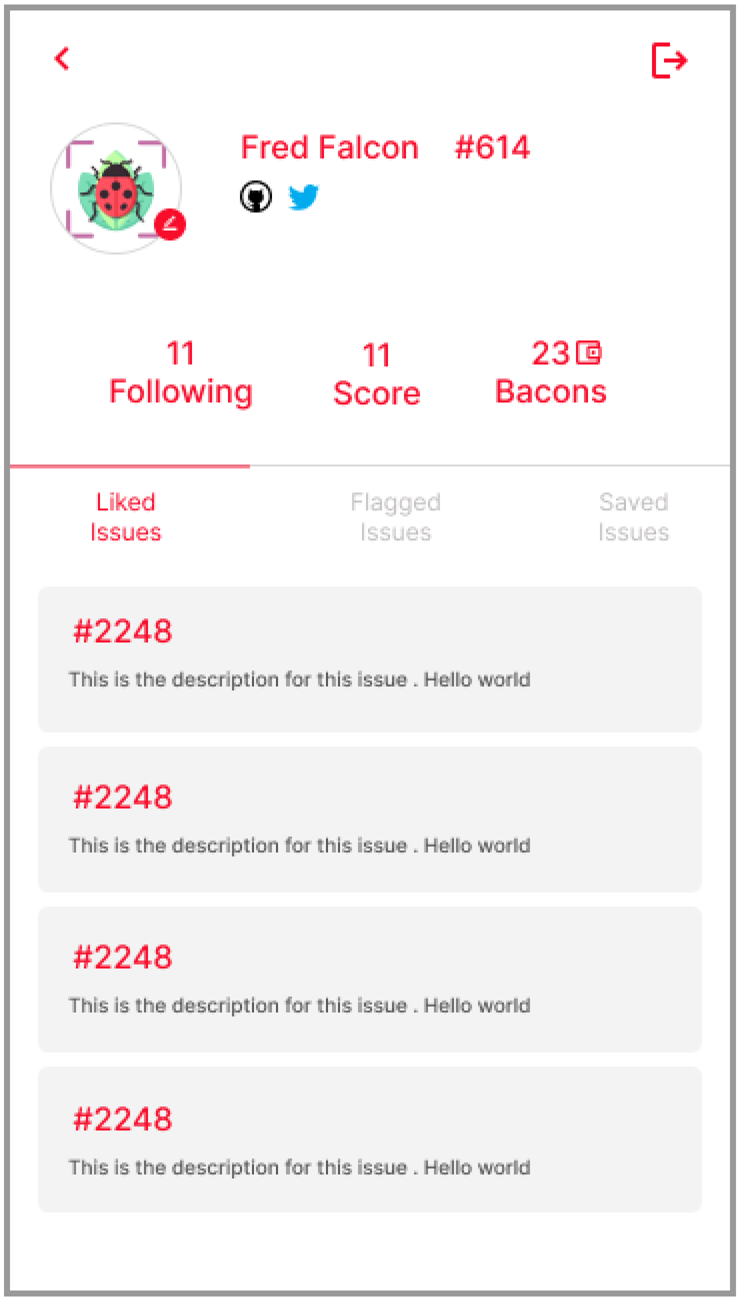
Ui is the same as **company-info** but only for users who is a admin of a company.



- **All sub-pages** of this page will be the same as the **sub-pages** of the **landing page**

- **Admin Privilege:** The admin will have the access to **close issues** and edit bug hunts on behalf of the company.

Profile Page:[Reform]


Fred Falcon #614
11 Following
11 Score
23 Bacons
Liked Issues
Flagged Issues
Saved Issues
#2248
This is the description for this issue . Hello world
#2248
This is the description for this issue . Hello world
#2248
This is the description for this issue . Hello world
#2248
This is the description for this issue . Hello world

- **UI Refactoring:** UI will be refactored to make it more appealing.

- **Wallet Integration:** The user will be able to view his bacon in the profile page.

Project Research and Brainstorming:

Different Approaches for Company Admin:

Single User Admin	Multiple Users as Admin but the user who created the company is super admin
This will not allow us to give one user access to make multiple companies.*	This will allow us to give one user multiple companies.
Also, the contribution from multiple people will be hindered*	Allows contribution by multiple users to bug hunt.
Easier Development.	The development will be difficult. (not a problem :))

Also, there is already a model created named **Company Admin** which is suited for this purpose.

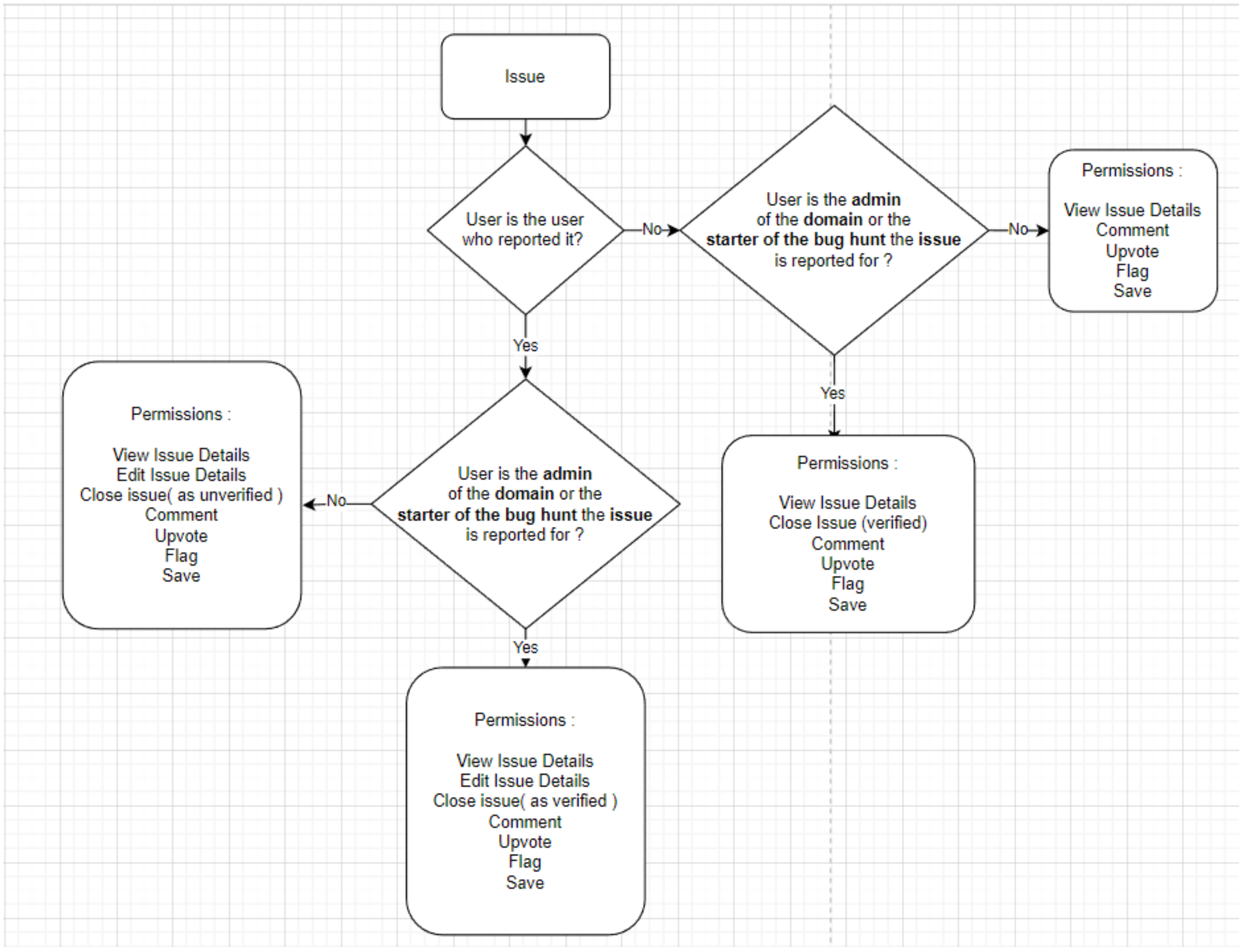
- The **admin field** in the **Company model** can be used for identifying the **super admin** who created the company and has the access to remove and add other **admins**.
- User can be shown a list of **companies** he is part of and upon clicking on one of them they will be **routed to** the **company dashboard**

* Let's say the company wants multiple people to evaluate issues of a bug hunt then they will have to share this id with multiple people then if a user is the admin of multiple companies there will be a threat that the people with whom the id is shared with approve the issues of bug hunt of other company.

Permission Logic:

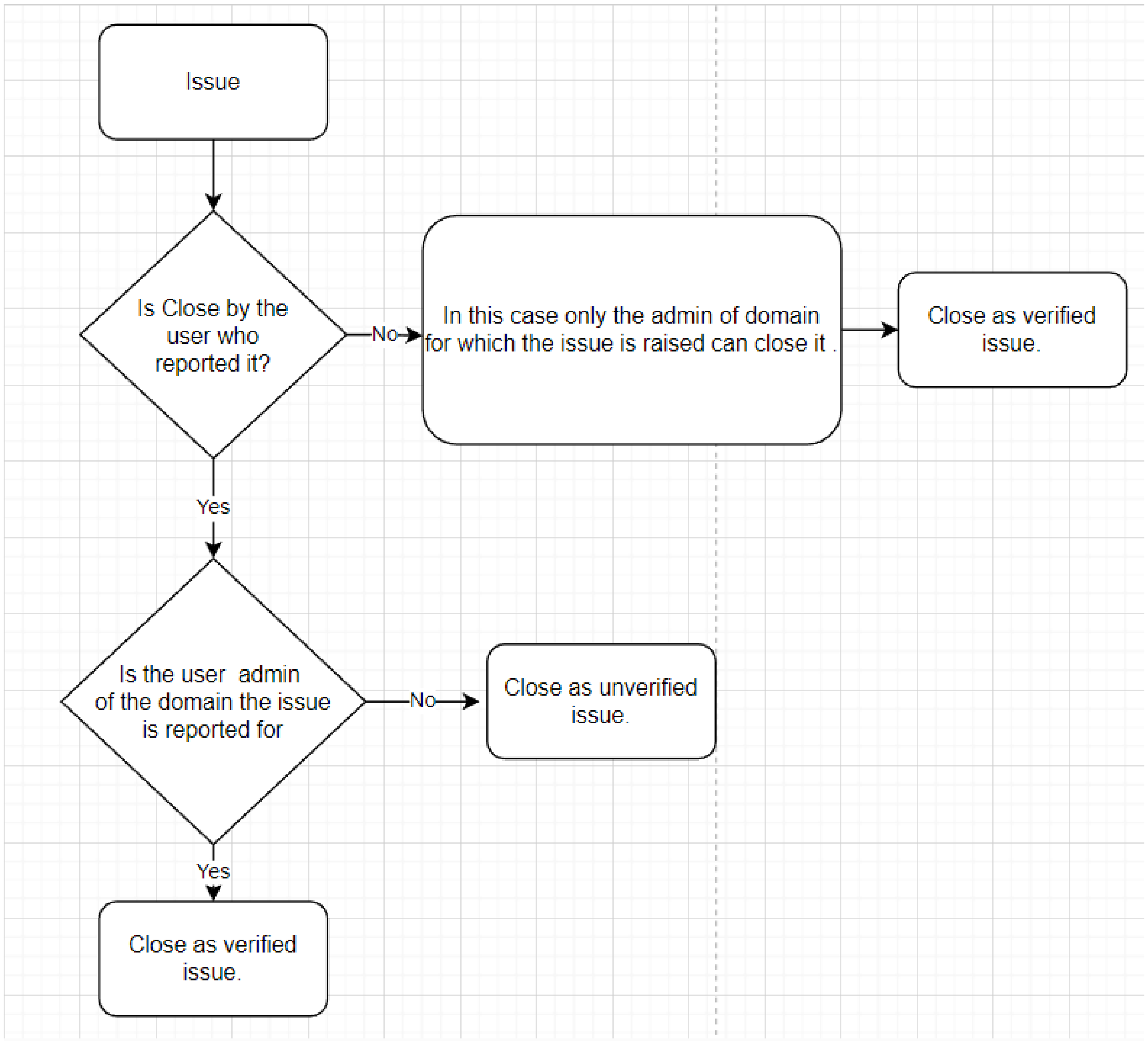
Issue or **Bug** is a central feature of our app. Following are the actions that can be performed by a user, on an issue, in our app.

- Comment
- Upvote
- Flag
- Save
- Close Issue (as verified)
- Close Issue (as unverified)
- Edit Issue


Issue
User is the user who reported it?
No
User is the admin of the domain or the starter of the bug hunt the issue is reported for ?
No
Permissions : View Issue Details Comment Upvote Flag Save
Yes
Yes
Permissions : View Issue Details Close Issue (verified) Comment Upvote Flag Save
User is the admin of the domain or the starter of the bug hunt the issue is reported for ?
No
Permissions : View Issue Details Edit Issue Details Close issue(as unverified) Comment Upvote Flag Save
Yes
Permissions : View Issue Details Edit Issue Details Close issue(as verified) Comment Upvote Flag Save

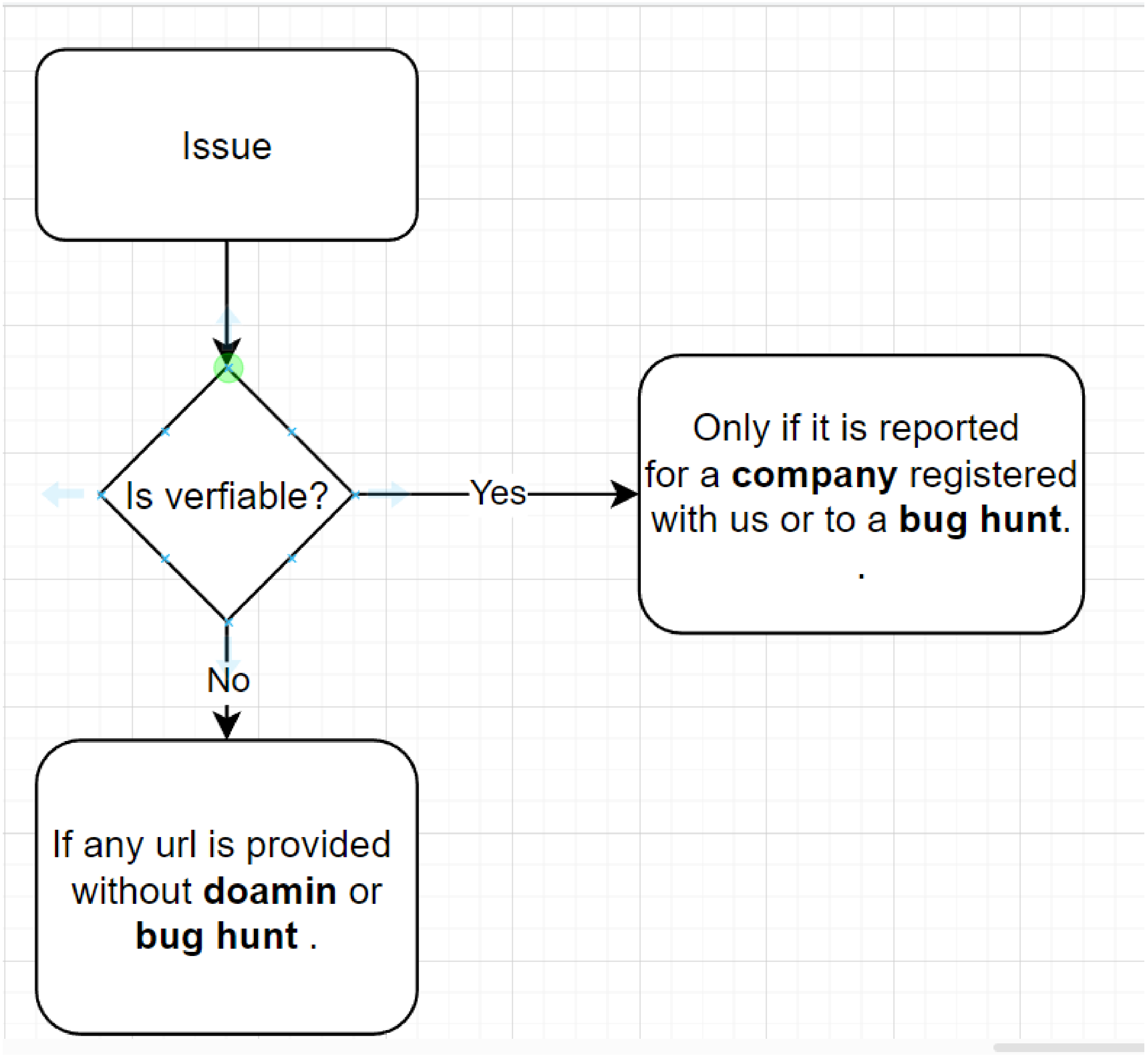
Permission Control Diagram for issues.

Closing an issue as a verified issue :

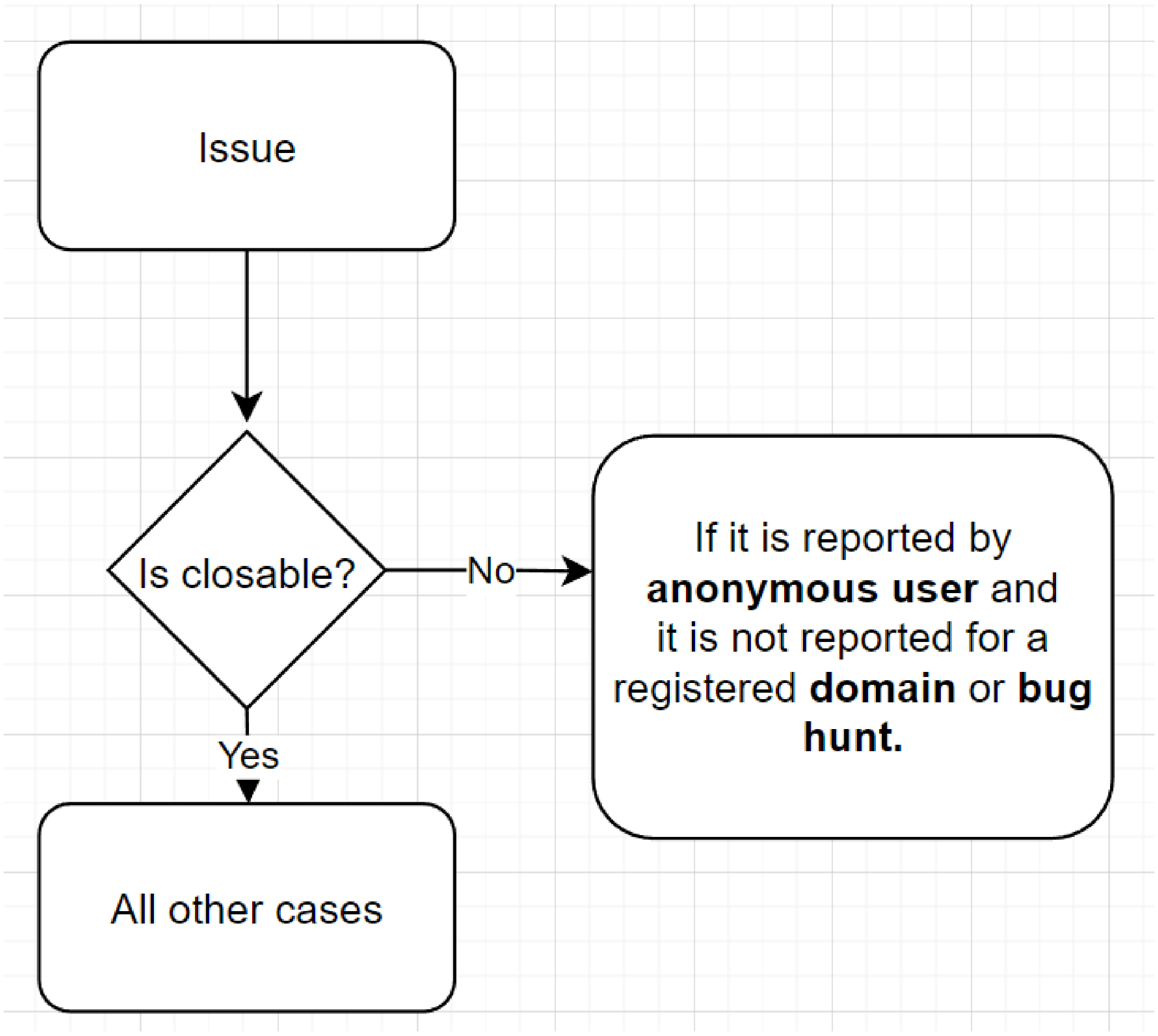

Issue
Is Close by the user who reported it?
No
In this case only the admin of domain for which the issue is raised can close it .
Close as verified issue.
Yes
Is the user admin of the domain the issue is reported for
No
Close as unverified issue.
Yes
Close as verified issue.

Limitations:

- An **issue** becomes **uneditable** if it was reported by an **anonymous user.**
- An **issue** becomes **not closable** if it was reported by an **anonymous user** and if it was not reported for a particular company or a bug hunt.
- An **issue** becomes **unverifiable** if it was not reported for a **company** or a **bug hunt**.
- A **closed issue** is not **editable**.


Issue
Is verfiable?
Yes
Only if it is reported for a **company** registered with us or to a **bug hunt**. .
No
If any url is provided without **doamin** or **bug hunt** .

Here is a **flowchart** of the condition in which an issue can be verified. Since the **verification action** can only be performed by a **company admin** or a **bug hunt starter** so it is required for an issue to be associated with a **bug hunt** or a **company** for being verifiable.


Issue
Is closable?
No
Yes
If it is reported by **anonymous user** and it is not reported for a registered **domain** or **bug hunt.**
All other cases

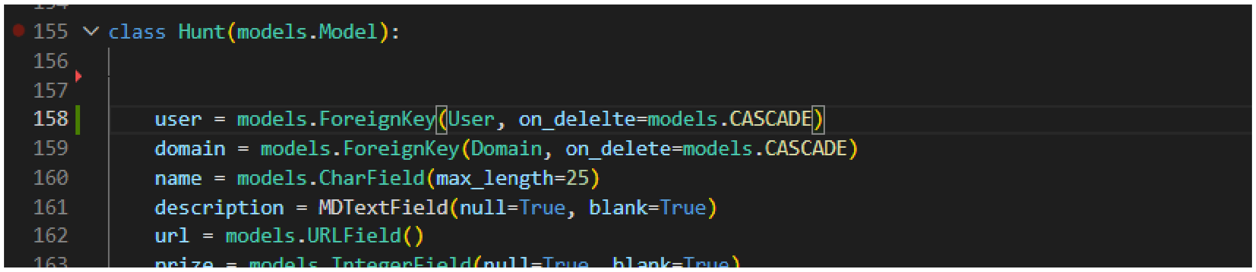
Here is a **flowchart** of the condition in which an issue can be closed. Since the **closing action** can only be performed by a **company admin**, a **bug hunt starter,** or the **user** who reported it so, it is required for an issue to be associated with a **bug hunt** or a **company** if reported by an **anonymous use**r for being closable.

Bug Hunt is another central feature in our app. Here is a flow for how it can be proceeded with in the App.

Both normal **users** and **users** who are company admins of a company will be able to start a bug hunt of their own however the **users** who are company admins will also be able to start a bug hunt on behalf of the **company**.

Model Field Required

- A user cannot independently start a bug hunt without having a company registered because then he will not be able to upload **results** or otherwise there will be security concerns for the API. So the solution is to introduce a **user** field to the Hunt Model This will have two benefits:
    - In the case of a **company** the **company admin** can be set as the user which will reduce queries for the database engine for permission check
    - It will allow us to implement the feature in which a normal user can start a bug hunt.



```
class Hunt(models.Model):


    user = models.ForeignKey(User, on_delelte=models.CASCADE)
    domain = models.ForeignKey(Domain, on_delete=models.CASCADE)
    name = models.CharField(max_length=25)
    description = MDTextField(null=True, blank=True)
    url = models.URLField()
    prize = models.IntegerField(null=True, blank=True)
```

Fields of **Hunt** model

Filters as proposed in my project are present on almost all screens of the app. Here is a list of all filters proposed in the App.

Django provides a very good filtering method but some filters can be difficult to implement like the company issue filter that can be over come by

- For Issues
    - User: By default, it will show all the issues
        - **Your Issues**: Issues Reported by you.
    - Status: By default, it will be set to all.
        - **Open Issues**
        - **Closed**

        - **Closed as verified**
    - Label: By default, it will be set to all kinds of labels
        - Any label present in the **API** currently, there are 8 labels i.e. **General, Number Error, Functional, Performance, Security, Typo, Design, Server down**
- For Bug-Hunts
    - User: By default, it will show all the bug-hunts
        - **Your BugHunts**: BugHunts started by you
    - Status: By default, it will be set to all.
        - **Active Bug-Hunts**
        - **Closed Bug-Hunts**
        - **Upcoming Bug-Hunts**: A published bug hunt in the future whose payment has been done or the is_published is true.
        - * For Your Bug-Hunts **Draft Hunt** filter will also appear.
- For Company Issues and Bug Hunts:

  All above-mentioned filters as well as:

    - **Domain:** In the **company-detail** page and the **company dashboard** page the user will be able to filter issues and bug-hunts on the basis of the domain of the company was reported or started for.

BACON Integration Implementation:

I propose to use the walletconnect_dart package for integrating the Flutter app with the meta mask wallet of the user.

We will be needing a **wallet address** field in the **UserProfile** model to keep track of his wallet address.

Payment Integration with PayPal:

I propose to use the django-paypal. So basically the user will be redirected to the paypal site inside our django payment instance and on completion of the payment our database will be updated.

API endpoints affected or created:

- api/v1/issue/
- api/v1/issue/<pk:id>/edit/
- api/v1/issue/<pk:id>/close/
- api/v1/bughunt/
- api/v1/bughunt/<pk:id>/results/
- api/v1/bughunt/<pk:id>/edit/
- api/v1/bughunt/<pk:id>/close/
- api/v1/bughunt/<pk:id>/leaderboard/
- api/v1/bughunt/<pk:id>/issue/
- api/v1/company/
- api/v1/company/<pk:id>/issue/
- api/v1/company/<pk:id>/bughunt/
- api/v1/company/<pk:id>/leaderboard/

Deliverables:

- **Mid-Term Evaluation Deliverables:**
  - Full Django API and Tests
  - Issues Page, BugHuntPage, and BugHuntDashboard and all filing screens which involve
    - Issue Detail Page
    - Issue Edit Page
    - Issue Reporting Page
    - BugHunt Edit Screen
    - Start Bug Hunt Screen
- **End-Term Evaluation Deliverables:**

  Full-fledged **app** and the corresponding **API.**

  - Django API and Tests
  - Company Info Page, Company Dashboard Page, Profile Page
  - BACON wallet Integration

Timeline:

Pre GSoC Period	
Before May 4	● Study the existing code. ● Clear doubts regarding the project ● Try contributing to issues in existing repositories.
Community Bonding Period	
May 4 - May 28	● Get to know more about the OWASP community. ● Participate in any ongoing discussions and take feedback and suggestion. ● Discuss Figma designs and update them.
Coding Period	
May 29 - June 15	● Django API and Test
June 15 - June 23	● Issue Screen ● BugHunt Screen ● Tests for the API
June 23-June 30	● Bug Hunt Info Screen ● Issue Detail Screen
July 1 - July 7	● Bug Hunt Dashboard Screen ● Bug Hunt Form Screen ● Start Bug Hunt Form Screen
July 8 - July 13	● Buffer Period for Mid-Term Evaluation
Evaluation Phase 1	
July 14 - July 21	● Company Info Page
July 21 - July 28	● Company Dashboard Screen
July 28 - Aug 4	● Profile Screen ● Wallet Integration with BACON
Aug 4 - Aug 11	● Fix anything left and code cleanup(if any required).
Aug 12 - Aug 21	● Buffer time to finish anything left according to the proposal (if any), or do some initial work on proposed future ideas.

Availability:

I would be easily available to dedicate around 30 hours a week. My college's summer breaks are for the entirety of May, June, and July. Other than this project, I have no commitments/ vacations planned for the summer. I shall keep my status posted to all the community members on Slack every week and maintain a transparent behavior on this project.

Why me?

With my experience working on several improvements for the BLT-Flutter app and my familiarity with the BLT codebase, I am confident that I already have a head start on my project. In addition, from my previous development experience, I have gained extensive experience working on various technologies, which have taught me the art of debugging for almost any issue that can arise during the project. Also, I am always eager to learn new technologies and am willing to do so if the need arises.

After GSOC:

I love building stuff and jamming over ideas, and after GSOC, I will have a deeper understanding of the BLT project. Since I would have already familiarized myself with the BLT project, I believe that continued contribution will provide me with even more opportunities to learn and grow. I would love to brainstorm in the community discussions and get newcomers in sync with the project. Also, I want to implement some future ideas like previous search and auto-completion as it will improve the user experience, and what's more fulfilling than user satisfaction. :).

Future Ideas:

- **End to End Tests for the App:** Tests are an important part of any application's credibility and ensure further development on a set standard. **Integration tests** can be written at once to ensure how different components interact with each other. Then widget tests can be written for widgets along with updating integration tests.
- **Previous Search:** The user will be able to view his previous searches. The data regarding this can be stored in the app itself. A package that can assist is input_history_text_field.

- **Image Caching:** Caching images can help the app load faster. A cache management system needs to be implemented for this purpose. Some packages which can be used are flutter_cache_manager, and cached_network_image.
- **Auto-complete domain or search:** A solution can be implemented in which the user gets suggestions for his search and form filling. A solution that will assist in making this feature more efficient is easy_debounce which will help in preventing multiple requests in an auto-complete operation.
- **TimeStamped Data for monitoring live bug hunts** and for **Activity Sections:** Timestamped data can be tracked and generated to show users a detailed analysis of their activity and companies an overview of their engagement.
- **Dark Theme:** Surely, an interesting feature for any app. But first, we need our app to cross the **MVP** stage.
- **React Frontend** for website**:** After this project, almost the entire **API** will be ready so we can easily move on to **API-based web apps.**
- **App Internationalization:** Only 15% of the world's population speaks English so in order to reach a greater audience, this feature will be required.
- **Caching in Django:** Caching will improve the latency of our backend.

References:

- https://github.com/OWASP/BLT-Flutter
- https://github.com/OWASP/BLT
- https://github.com/OWASP/BLT-Bacon
- https://owasp.org/www-community/initiatives/gsoc/gsoc2023ideas
- https://owasp.org/www-community/initiatives/gsoc/gsoc2023
- https://owasp.org/www-community/initiatives/gsoc/gsoc_sat