Instabug Task

[1] Description:

Instabug wants to build an app for tracking customer reviews for Instabug's services, The app is currently built using VueJS-2, AngularJS, and NgVue (Bridge between AngularJS and VueJS).

The app mainly consists of a single page showing the reviews, and a not-found (404) page to handle unknown routes. You will be given below a list of requirements to make our app ready for release and get more insights about Instabug's services from our customers. 🎉

[2] Installation & Instructions:

- Make sure to use node 14.17.0, you could use NVM to install it if you don't have it already.

- Make sure to use yarn in installing and running the project.

- Decompress the provided file and create a GitHub repository for the task but YOU HAVE TO MAKE IT PRIVATE and give access to the following list of accounts as collaborators (Check this link to know how to add them)
    - AbdallahHemdan, aelfangary, majidzeno, mohamedelhosseiny, abdrahmansoltan, geemeows, sirSayed98, bolamsydhom

- Run `yarn install` to install the dependencies.
- Run `yarn start` to run the project.

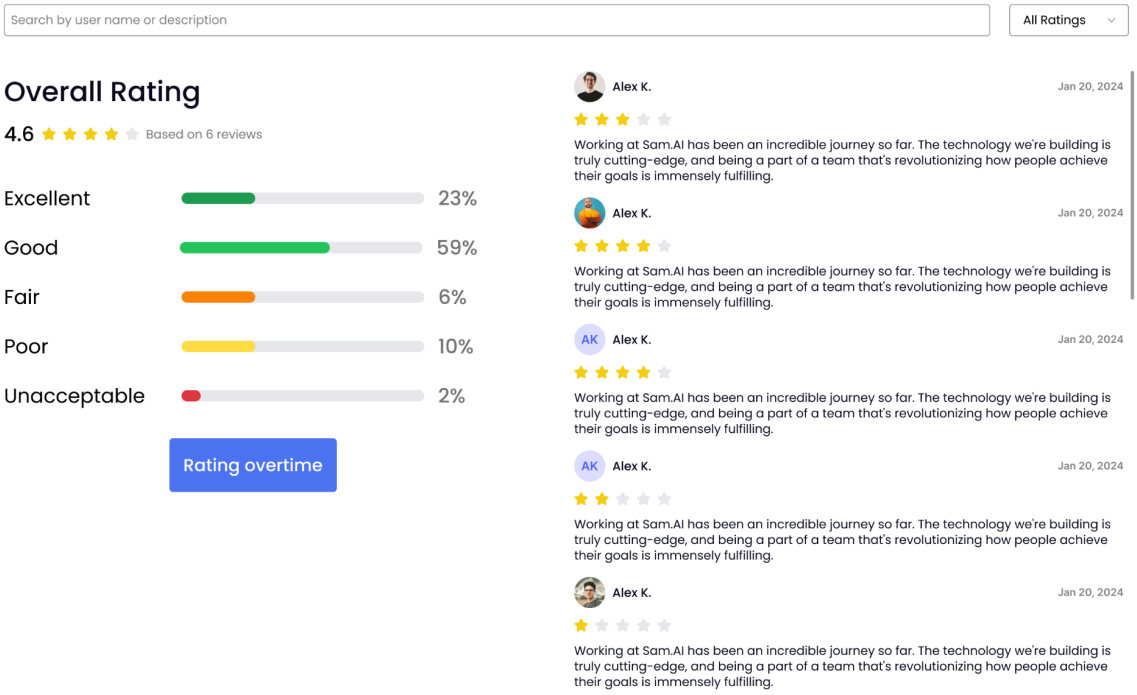
- Create a single Pull Request (PR) for each requirement of the list below where its title is the task title.

- All the requirements have design screens related to them in this FIGMA file.

- You will find the Vuex store, and AngularJS service, feel free to choose whatever you see suitable more in terms of state fetching and management.

[3] Requirements:

Here is a general overview of the expected output of the page (more details in the [figma file provided]).



Requirement #1: Reviews List

Display the list of review as shown in the attached screenshot given the following notes:

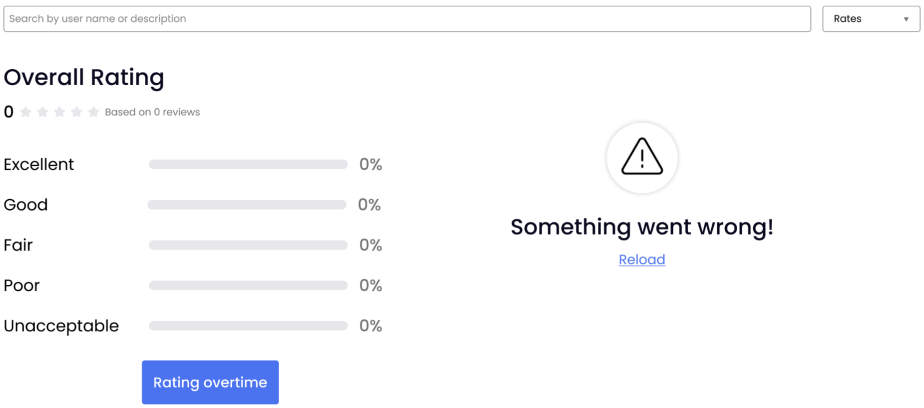
1. Reviews list is an infinite scroll list, when the user reaches to the end of the fetched list (user already viewed the last set of reviews) you should be getting more items.

2. Page size is 10 reviews and here is the endpoint for getting the reviews list https://frontend-task.instabug-dev.com/api/web/reviews?page=x

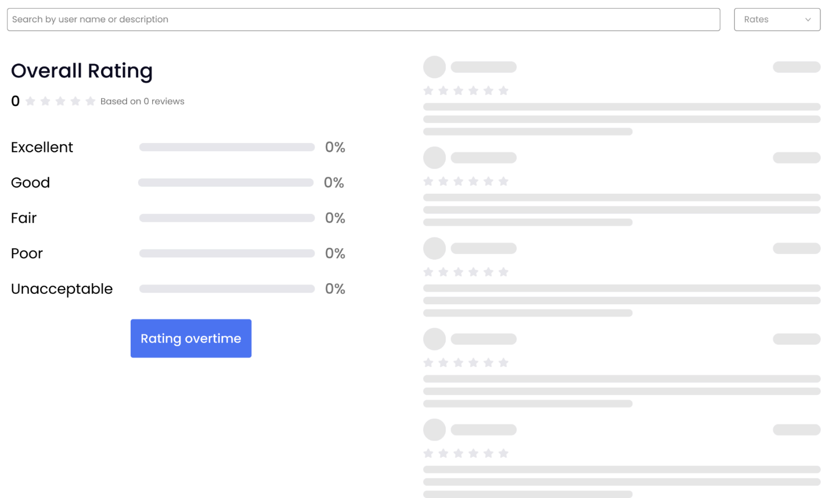
3. `has_next` field in the above endpoint indicates whether we have new page of reviews or not.

```javascript
{
  "reviews": [...],
  "current_page": 1,
  "has_next": true
}
```

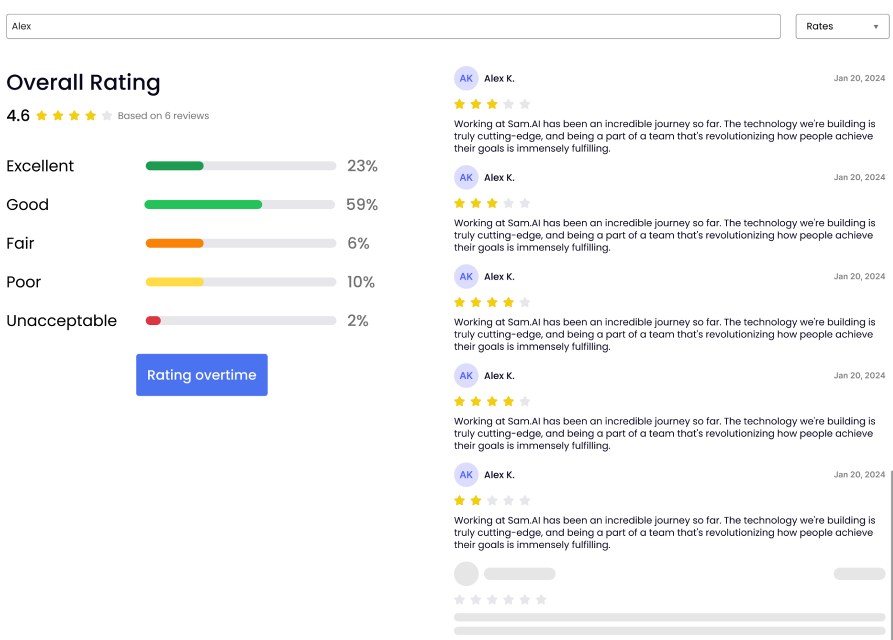
4. If there is an error fetching the reviews list, an error state should be displayed, as per the provided design.



5. On the initial page load (fetching reviews for the first time), a loading state (skeleton) should be displayed.



6. When user scroll to end of list and you are fetching new data, a loading state (skeleton for the new item should be displayed).



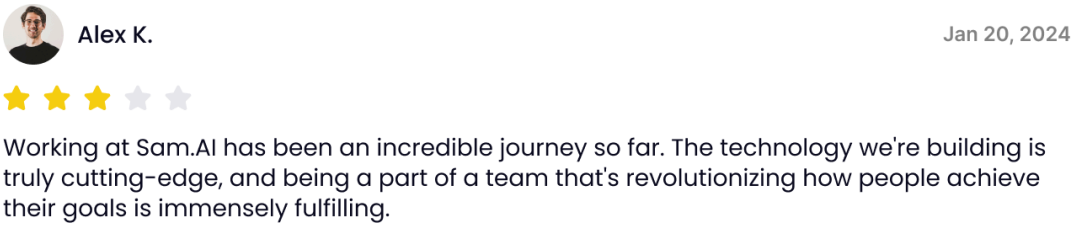
7. Review Item should contain the following information as shown in the design.

    a. **Profile picture**: If present, display image. Otherwise, display 2 letters of user name (user_id) with a random background color.

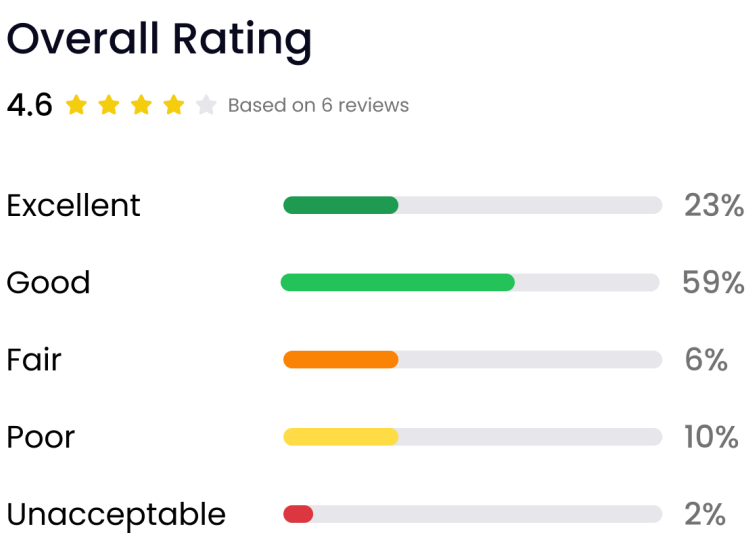
    b. **Rating stars**: Display filled stars according to the given rating.

    c. **Review**: Display review as given.

    d. **Date of review**: The date format should be displayed in the `MMMM d, yyyy` pattern, where `MMM` is the month name, d is the day of the month without a leading zero for single-digit days, and yyyy is the four-digit year.



Requirement #2: Reviews Summary

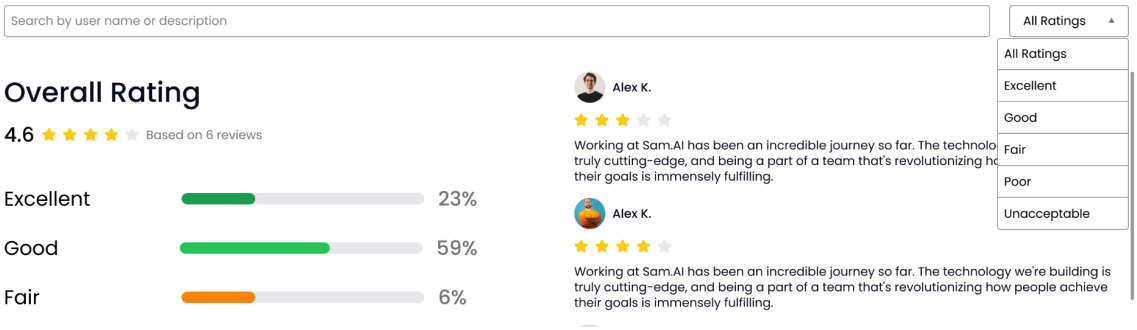


You should add a new section for review summary similar to the screenshot above.

1. **Reviews number**: Number of fetched reviews so far.
2. **Overall rating**: Calculated by averaging all fetched ratings.
3. **Rating bands**: Summarize the percentage of each rating band of reviews.

- **Notes**:
- The reviews number, overall rating, and rating bands should be dynamically changed when you scroll and fetch more reviews.

Requirement #3: Reviews Filtration

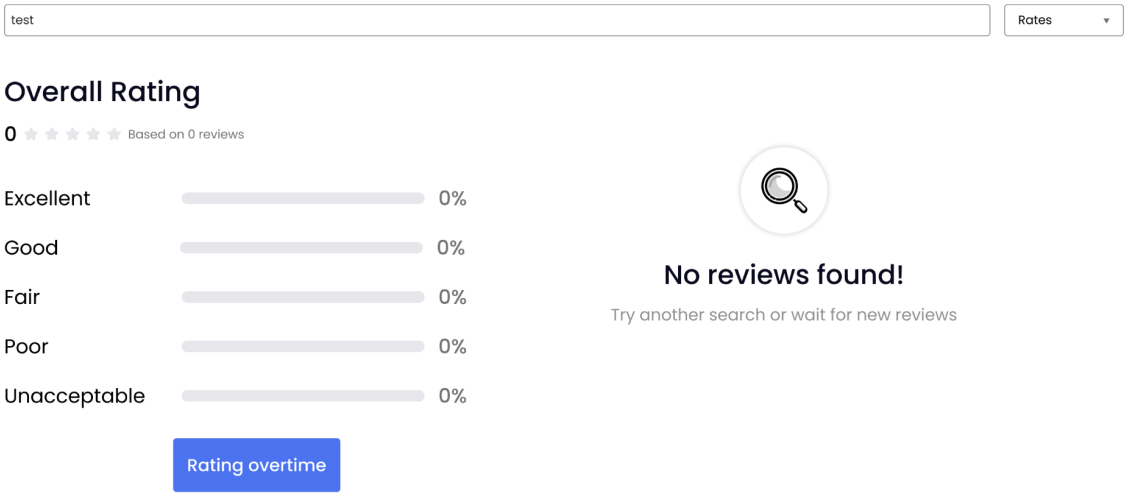


You should add two new components to the top of the page as follows:

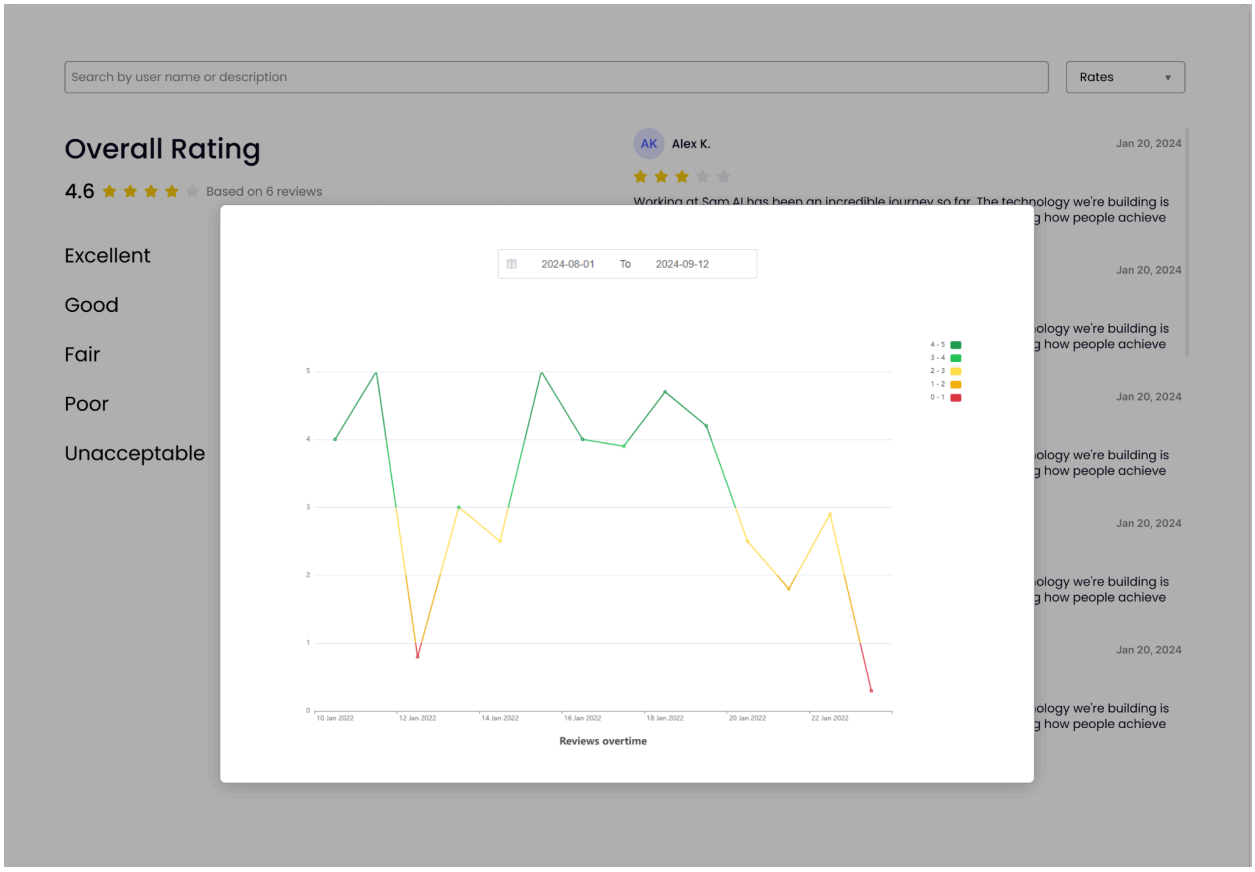
1. **Search input box**: Match the search value with any review content or username.

2. **Ratings dropdown**: Match the reviews that have this specific rating.
    a. Excellent = 5
    b. Good = 4
    c. Fair = 3
    d. Poor = 2
    e. Unacceptable = 1

- **<u>Notes</u>:**
- All filtering should be applied on the *<u>fetched</u>* reviews only.
- While filtering, you shouldn't apply the infinite fetching (disable fetching new reviews).
- "All Ratings" should be the default filter on the page and no search key applied.
- If there are no reviews that match the applied filter (Ratings or Search), you should be showing the empty state.



Requirement #4: Reviews Overtime



You should add a new modal (dialog) component to show the ratings of our app overtime given the following notes:

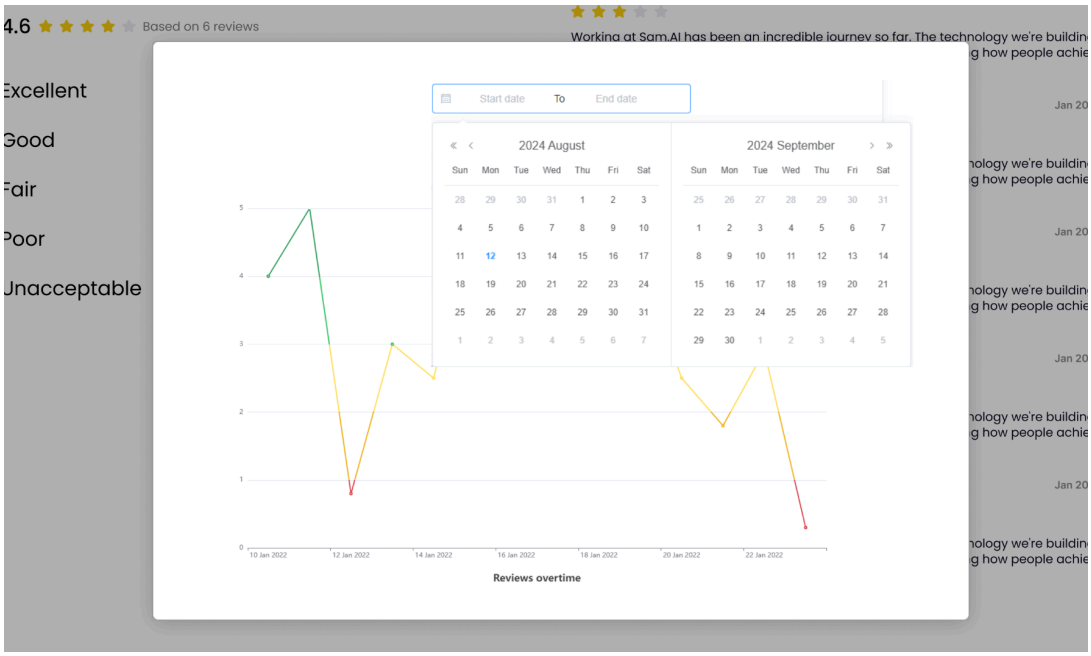
- We already added a chart component built with the echarts library for you inside the path of *`/app/vue-components/reviews-overtime-chart.vue`*.
- You could use any library for the modal or implement it on your own. Here is an example for UI library for this https://element.eleme.io/#/en-US/component/dialog
- The provided component has a static date, you need to fetch the data dynamically from this endpoint https://frontend-task.instabug-dev.com/api/web/reviews_overall
- The endpoint will return you all the available data overtime, you shouldn't provide it with a date range.

JavaScript

```
{
  "data": [
    {
      "date_ms": 1722470400000,
      "value": 3.24
    },
    {
      "date_ms": 1722556800000,
      "value": 4.7
    },
    ...
  ]
}
```

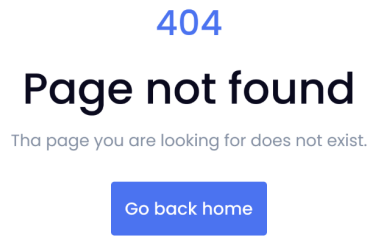
- You should add a date picker to filter the data points based on the selected date range from the user (All date range filtration should be a frontend side).

- You could use the html default date range input or any library for the date range or implement it on your own. Here is an example for UI library for this https://element.eleme.io/#/en-US/component/date-picker#date-range



Requirement #5: Not Found Page

We already add a handler for not found routes, we are just asking you to style it and make it similar to the design provided.



Requirement #6: Bonus

You will find cypress installed, automate these two scenarios:

1. **1st scenario**
    a. Open home page
    b. Mock/Stub the reviews request
    c. Assert on the first review that all its data are shown correctly
    d. Assert on the aggregations are shown correctly based on the mocked reviews

2. **2nd scenario**
    a. Open home page
    b. Mock/Stub the reviews request
    c. Scroll down to the reviews list
    d. Validate that the skeleton is show

General Requirements

1. **Design fidelity**: Aim to follow the design as closely as possible. All elements in the design should be present, using the specified text color, font size, font weight, spacing, dimensions, etc.

2. **Performance optimization**: Code for fast load times with efficient CSS and JavaScript techniques, mainly optimizing for the initial page load and initial bundle size.

3. **Meaningful namings**: Make sure to use meaningful names for commit messages, variables, function names, and html classes.

4. **Code cleanup**: Make sure your final code is completely free of any debugging statement or console logs and so on.

5. **Accessibility and semantics**: Follow best practices for web accessibility, such as using semantic HTML and ARIA roles where necessary and using proper alt tags for images.

6. **Don’t be WET**: Don’t “Write Everything Twice”.