Information Management Group

Summer Project (the-year-which-must-not-be-named)

For developers: backend, frontend

Overview

You are very much familiar with the flow of launching an app at IMG. In this flow, Testing is one of the most crucial phases. Google Sheets are not very handy in the efficient management of the bugs; It creates a lot of hassle for the team as well as for reporter to explain the bug without any screenshots and proper details. As a part of your summer assignment, you have to make a full-fledged app having the functionality of an efficient bug tracking platform. The basic features are mentioned in the following document, but you are free to tinker about their implementation and other features. The end goal is to develop an in-house bug tracking platform ironed according to our needs.

Goals

1. **Account creation and log in:** The user must be able to register and log in using the new Channel i OAuth (https://omniport.readthedocs.io/en/latest/how_to/use_oauth2.html). Only the IMG Members (Maintainers) [The OAuth will provide this data] must be allowed to use your app.
2. **Authorisation:** There are two roles your users must be categorised in namely, Admins(should not be made by changing the default admin panel of Django), Normal Users. Normal users can do normal tasks like Creating a Project, reporting a bug, Commenting on the reported bug etc. Admin can do whatever Normal Users and can disable the users (A disabled user shouldn't be able to log in again), change the role of a user from Normal to Admin and vice-versa. Admin can perform any action in the app.
3. **Add/Edit/Delete Project:** Anyone can create a new project in the app to report the bugs for that project. Some required fields for the project can be a name, a wiki, and team

members. Wiki can be used to add the basic test instructions there only. The creator of this project can now add more team members in the project. Now, the creator, team members and the admin can edit the project details (Wiki, team members and other editable fields) and even delete the project.
Suggestion: You may use RichTextField for the Wiki.

4. **Add/Edit/Delete New Issue:** On the home-screen user can see a list of all the available projects, and on selecting a project can see all the reported bugs. User can report a new issue in the project. The issue must contain a heading, optional description, optional media upload option, tags (enhancement, bug, UI/UX, etc.), status (pending, resolved, to be discussed etc.). The Issue reported, the project team member, the admin must be able to change the status and delete the issue. Besides, the admin and team members can assign the issue to one of the team members. The description, heading tags are not editable once created.
   Suggestion:
   You may use RichTextField for the description of the bug.
   You may use colours to show status, show tags on the list page, etc., try your best that your project will be the one that gets deployed in the lab.

5. **Commenting:** Anyone must be able to comment under the reported issue. This can help in the discussion on the issue. The commenting system must be real-time (with the help of Websockets).

6. **My Page:** There must be a “My page”, where a user can see it’s assigned and reported issues.

7. **Members (For Admins only):** The Admin view allows to see the whole list of members with basic details and perform tasks mentioned above for the admin.

8. **Email notifications:** It is really easy to integrate email notifications at the backend. There must be emails for the important actions to the apt users with required details. For example to the team members when an issue is reported in their project, to the team members and the issue reporter when there is a comment on the issue or the status is changed etc.

Specifications

1. **Database:**
   Use either PostgreSQL or MySQL for storing your data and not SQLite. SQLite is a file system database and hence is not suitable for storing data in production. Make sure to do this.

2. **API with Django Rest Framework:**
   Tech stack: Python; Django, Django REST framework
   This will be the framework that should be used to write the backend part of your application. You can find all the information and tutorial about this framework at this link. You need to use **ModelSerializers** and **ModelViewSets** for making your API. Keep this in mind when developing the app. The permissions must be in a separate file named *permissions.py* and try to reduce code repetition.

3. **Frontend with React:**
   Tech stack: JavaScript (JSX); ReactJS
   For the frontend of the application, you will be using React.js. Since most of you are new to it, it is recommended to first learn the basics of React by going through the official documentation. The official documentation is more than enough for you to be able to complete this application. The benefit of using React over conventional HTML and javascript will be clear once you start learning and using it. Everyone loves good design, but not everyone can design good (at least me, according to my experience with portfolios :p). Bootstrap is for kids; you can check out Semantic UI React or Ant Design of React to make your life slightly more comfortable.

Milestones

Since this is your first full fletched project, it becomes essential that we evaluate and help you at times. Milestones will also help you to plan things and work more efficiently. Although we don't demand as well as expect you to spend your entire "holidays" on this, it does require some amount of commitment. Below are tentative dates for you to report your progress to your junior mentors(second years). In case of doubt, you can contact any of second or third years.

1. **Database structure and screens(Till 30th April Midnight):**

   This is the very first and foremost step in your app development. You need to list down all the features in a document. Next, decide the database structure. For this, it is advisable to make ER diagrams; a verbal explanation will work as well :p. For better understanding, you can make the hand-drawn layout of the screens that you plan to build. All this has to be shared and will be reviewed by second and third years.

2. **API's / Backend(Till 20th May Midnight):**

   Now comes the backbone of your app, the API's :'). Read up on the Django Rest Framework. Needless to say, your code should be git tracked, and you will be evaluated on the progress that you make coding your API. It is expected that you follow all the specifications mentioned above strictly.

3. **Frontend(Till 17th June Midnight):**

   The most interesting and probably the most challenging part of your app. Once you are ready with your APIs, you should get started with the frontend. You are free to decide the design of your apps do make sure that it is simple to use. The hand-drawn screens made in the 1st part will come in handy and try to build on them. For inspiration, you can look at various bug tracking apps online.

Progress, Confusion and Resolution

Confused..!!! No worries, you can always reach out to second or third years any time.

To start with the assignment, you are required to make a GitHub repository and add your mentors as collaborators so that they can monitor your progress.

If you have any doubt in understanding what needs to be done or any confusion related to the tech stack, feel free to contact us.

Stay Safe and Happy Programming!!