Git Light: Project Specification

Team 18
Zihao Cao | Andrew ID: zihaocao
Zhengpei Li | Andrew ID: zhengpel
Liangyi Zhao | Andrew ID: liangyiz

Project Backlog

1. Register and Log in

We support user authentication system. User log in and register are support by the website. We further support forget password functionality, which allows user use email authentication whenever they forget their password.

2. Issues and Discussion

In a repository, one may come up with an issue regarding bugs, improvements or suggestions.

Further, in an issue, one may follow up and join this discussion. The discussion can be about solving the questions others put up with, approving someone's opinion and come up with a following question.

3. User Profile Page

In the profile page, a profile picture can be uploaded and biograph is supported. Also, all repos belong to a user will be shown, and redirect to the repo page whenever you click on one repo. Also, you can visit other's profile to see their profile picture, their bio and the repositories they have.

4. Basic GitHub Functions:

Display git history, changes between previous version and current version, and branches.

5. Git Operations:

Instead of using git server to create a repository, a branch, and clone and fork repository, one may finish these instructions via our website. To create a repository, click on "Start a new project" in the index page and enter any repository name and click on "Create Repository". To create a branch, click on the "Create a new branch" in the repository page. To clone a repository, click on the button "Clone" int the repository to get URL of the repository.

The First Sprint Backlog

1. Sign in and Register

The pages for those who got the account of our website to login. Or for fresh user to register. Email verification will be accomplished.

2. Dashboard Page.

After login or registered, the web show the dashboard of the user, where user can redirect to other pages or check their status.

3. Profile.

User biograph update and profile picture support.

4. Basic GitHub Functions.

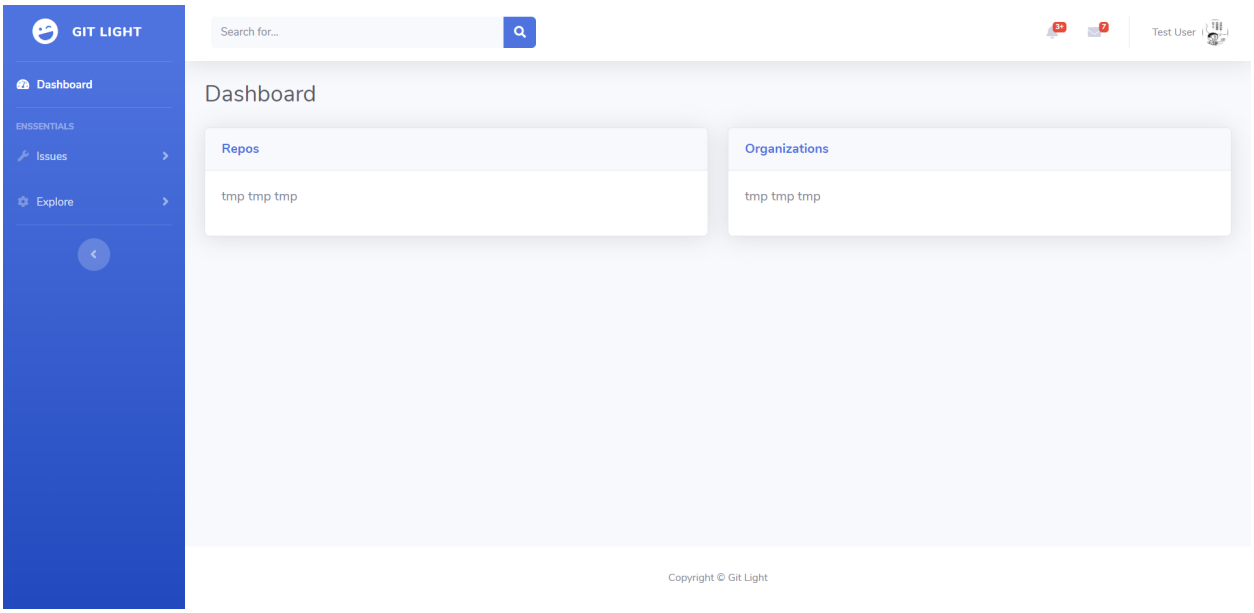
Display git history, changes between previous version and current version, and branches.

5. Issues and Discussion

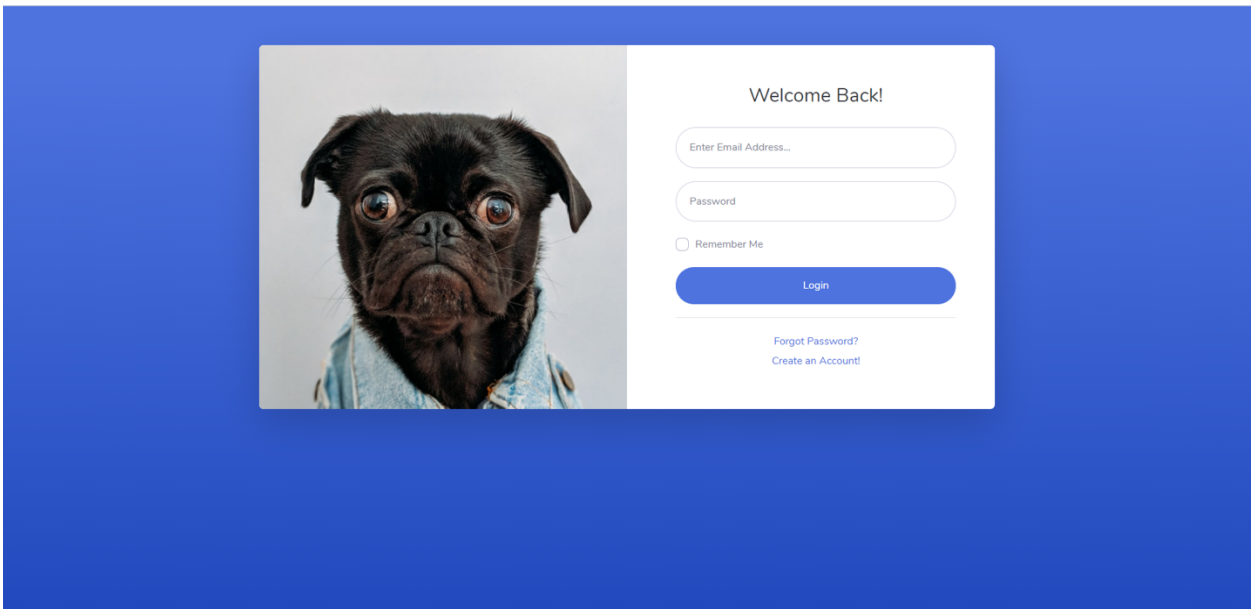
Issue come-up function and discussion function on issue page.

HTML Mock-ups

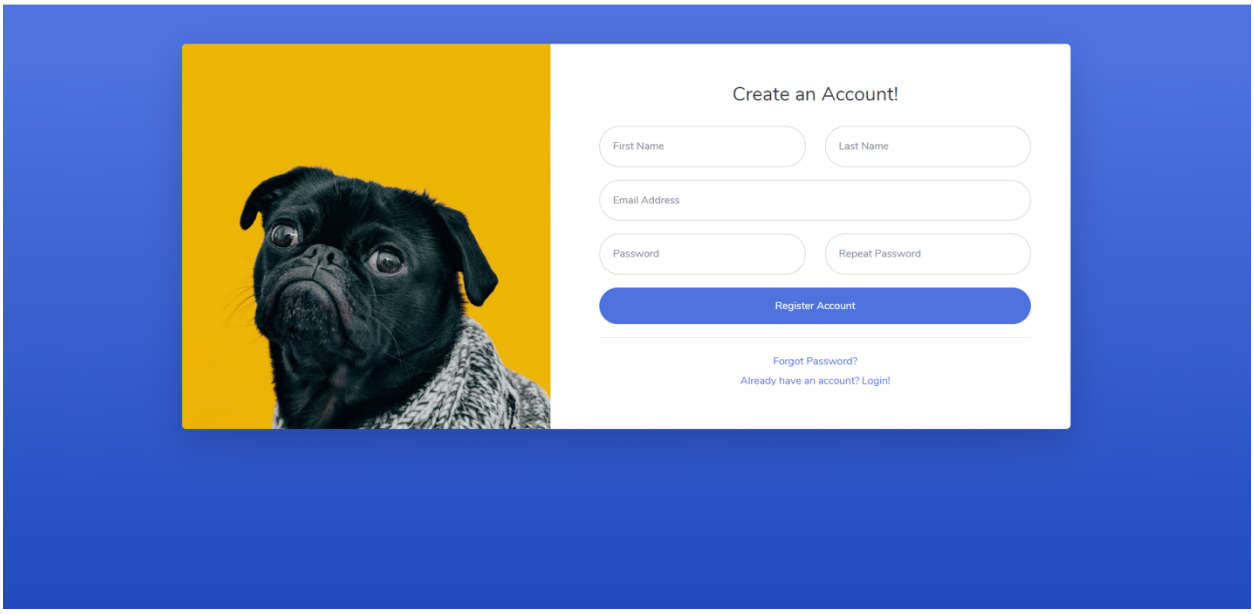
We are using bootstrap and a template called SB Admin 2 in our front-end design. Currently we have made index page, login page and profile page. These pages are as follows.



Index page



Login page



Register page

Here are some descriptions for these pages.

Login page:

Clicking “create an account” will redirect to register page;

Clicking “forget password” will use email verification to reset password;

Register page:

Clicking “already have a account” will redirect to login page;

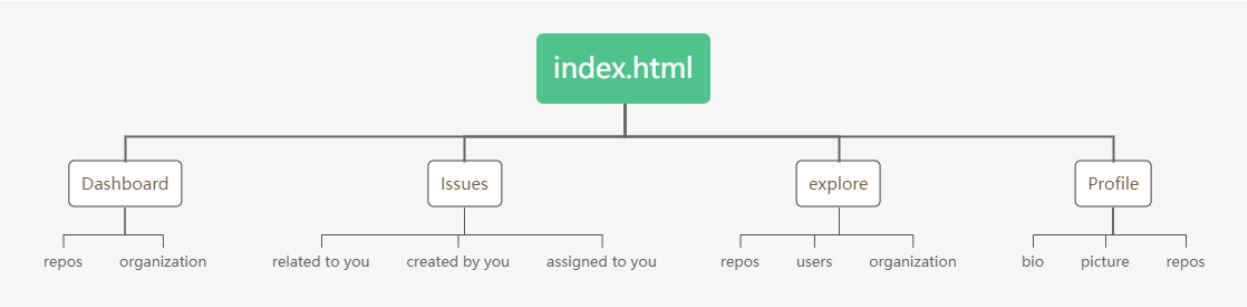
Clicking “forget password” will use email verification to reset password;

Index page:

Clicking the users from explore section will redirect to the user’s profile page;

Clicking the any repo from Issue section will redirect to the repository's page;
Clicking the any issue from Issue section will redirect to the repository's page which this issue belongs to;
Clicking the organization from explore section will redirect to the organization's page, which will list all users and repositories this organization have;

Structure of the project is as follows.



Structure of the webpage

Dashboard:

Repos: repositories that the current user have;
Organization: organization that the current user belongs to(optional);

Issues:

Related to you: issues that related to your repositories or the repositories that you are participated in;
Created by you: issues you come up with;
Assigned to you: issues that the leader of the organization assigned to you (optional);

Explore:

Repos: all repositories;
Users: all user;
Organization: all organization (optional);

Profile:

Bio: biograph of a user
Picture: profile picture
Repos: all repos that a user have

Data Models

To make our website working, we will need a data model issue to record every issue that are come up with user; a profile model to record each user's profile; a discuss model to allow users to discuss under an issue; an organization model that allows user to join an organization (optional). The following is our best guess to the models, and all of them are wrote in Django model form (a pseudo version, may contain bugs).

Note that we don't have a data model to record repositories. Because that we will use Dulwich as a bridge between python and git server. We are still working on it. Those models which is related to repositories may need further modification as a consequence.

We will have an authentication model for users to register and login, which is basically a rip-off of what we have finished in social media homework.

```
class LoginForm(forms.Form):
    username = forms.CharField(max_length=20)
    password = forms.CharField(max_length=200, widget=forms.PasswordInput())

    def clean(self):
        cleaned_data = super().clean()

        username = cleaned_data.get('username')
        password = cleaned_data.get('password')
        user = authenticate(username=username, password=password)
        if not user:
            raise forms.ValidationError("Invalid username/password")
        return cleaned_data

class RegistrationForm(forms.Form):
    username = forms.CharField(max_length=20)
    first_name = forms.CharField(max_length=20)
    last_name = forms.CharField(max_length=20)
    email = forms.CharField(max_length=50,
                            widget=forms.EmailInput())
    password = forms.CharField(max_length=200,
                               label='Password',
                               widget=forms.PasswordInput())
    confirm_password = forms.CharField(max_length=200,
                                       label='Confirm password',
                                       widget=forms.PasswordInput())

    def clean_confirm_password(self):
        """Handle password mismatch."""
        cleaned_data = super().clean()
        password = cleaned_data.get('password')
        confirm_password = cleaned_data.get('confirm_password')
        if password and confirm_password and password != confirm_password:
            raise forms.ValidationError("Passwords did not match.")

        return cleaned_data

    def clean_username(self):
        """Handle mulitiple usernames"""
        username = self.cleaned_data.get('username')
```

```
        if User.objects.filter(username__exact=username):
            raise forms.ValidationError("Username is already taken.")

        return username
```

We will also need a data model to allow user to come up with issues under each repository. Further, discussion under issue will be supported.

```
class Issue(models.Model):
    user = models.ForeignKey(User, default=None, on_delete=models.PROTECT)
    issue_text = models.CharField(
        max_length=10000)
    issue_under_repo = models.ForeignKey(
        'Repo', default=None, on_delete=models.PROTECT)
    issue_assigned = models.ManyToManyField(User)

class Disscuss(models.Model):
    user = models.ForeignKey(User, default=None, on_delete=models.PROTECT)
    discuss_date_time = models.DateTimeField()
    discuss_text = models.CharField(
        max_length=10000)
    post_under = models.ForeignKey(
        'Issue', default=None, on_delete=models.PROTECT)
```

Also, each user will have his/her own profile page.

```
class Profile(models.Model):
    """Profile class specifies a user's profile"""
    user = models.ForeignKey(User, default=None, on_delete=models.PROTECT)
    biotext = models.CharField(max_length=10000)
    profile_picture = models.FileField(blank=True)
    content_type = models.CharField(max_length=50)
    repo = model.ManytoMany('Repo')
```