Kristy Gray

Part A

DRF Project
Submission Part 2 of 3

Project Requirements

Project: Crowdfunding

Project Description

Kickstarter, Go Fund Me, Kiva, Change.org, Patreon… All of these different websites have something in common: they provide a platform for people to fund projects that they believe in, but they all have a slightly different approach. You are going to create your own crowdfunding website, and put your own spin on it!

Project Requirements

Your crowdfunding project must:

- ✓ Be separated into two distinct projects: an API built using the Django Rest Framework and a website built using React.
- ✓ Have a cool name, bonus points[1] if it includes a pun and/or missing vowels. See https://namelix.com/ for inspiration.
- ✓ Have a clear target audience.
- ✓ Have user accounts. A user should have at least the following attributes:
    - ✓ Username
    - ✓ Email address
    - ☐ Password
- ☐ Ability to create a "project" to be crowdfunded which will include at least the following attributes:
    - ✓ Title
    - ✓ Owner (a user)
    - ✓ Description
    - ✓ Image
    - ✓ Target amount to fundraise
    - ✓ Whether it is currently open to accepting new supporters or not
    - ✓ When the project was created
- ✓ Ability to "pledge" to a project. A pledge should include at least the following attributes:
    - ✓ An amount
    - ✓ The project the pledge is for
    - ✓ The supporter/user (i.e. who created the pledge)
    - ✓ Whether the pledge is anonymous or not
    - ✓ A comment to go along with the pledge
- ✓ Implement suitable update/delete functionality, e.g. should a project owner be allowed to update a project description?
- ✓ Implement suitable permissions, e.g. who is allowed to delete a pledge?
- ✓ Return the relevant status codes for both successful and unsuccessful requests to the API.
- ✓ Handle failed requests gracefully (e.g. you should have a custom 404 page rather than the default error page).
- ✓ Use Token Authentication.
- ☐ Implement responsive design.

Additional Notes

No additional libraries or frameworks, other than what we use in class, are allowed unless approved by the Lead Mentor.

---

[1] Bonus points are meaningless.

Project Submission Details

Note that while this is a crowdfunding website, actual money transactions are out of scope for this project.

MVP Plan Submission

Please submit the following:

- ☑ The title of your website.
- ☑ A paragraph detailing the purpose and target audience for your website.
- ☑ The features your MVP will include.
- ☑ At least 3 additional features you would like to include if you reach your MVP.
- ☑ An API specification.
- ☑ A database schema.
- ☐ Wireframes. **mood-board instead**
- ☑ Colour Scheme.
- ☐ Heading and body font(s). **mood-board instead**

Part A Submission

Please submit the following:

- ☑ A link to the GitHub repository containing the code for your project.
- ☑ A link to the deployed project.
- ☑ A screenshot of Insomnia, demonstrating a successful GET method for any endpoint.
- ☑ A screenshot of Insomnia, demonstrating a successful POST method for any endpoint.
- ☑ A screenshot of Insomnia, demonstrating a token being returned.
- ☑ Step by step instructions for how to register a new user and create a new project (i.e. endpoints and body data).
- ☑ Your refined API specification and Database Schema.

Part B Submission

Please submit the following:

- ☐ A link to the GitHub repository containing the code for your project.
- ☐ A link to the deployed project.
- ☐ A screenshot of the home page.
- ☐ A screenshot of the project creation form.
- ☐ A screenshot of a project with pledges.
- ☐ A screenshot of the resulting page when an unauthorized user attempts to edit a project.
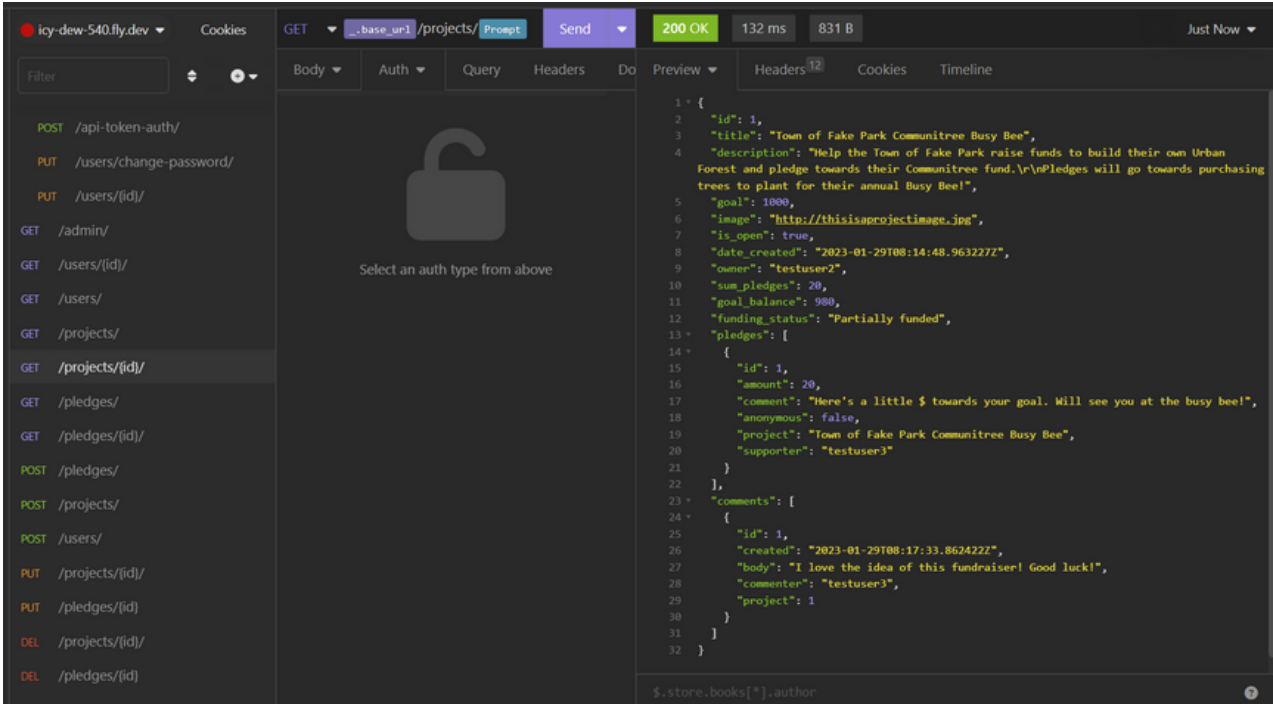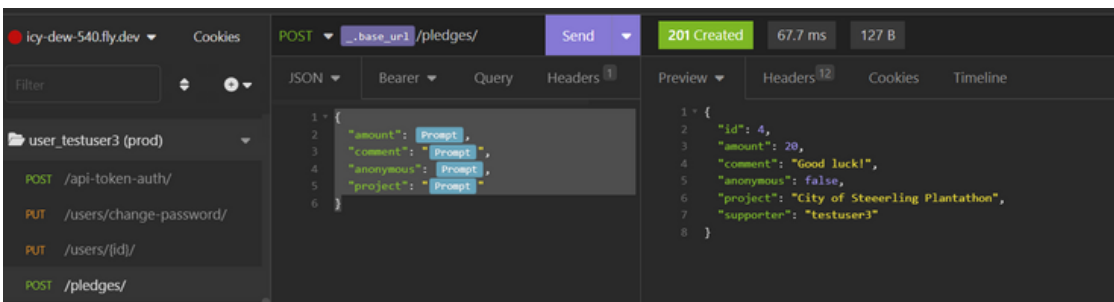
GitHub Project Repo

https://github.com/SheCodesAus/she-codes-crowdfunding-api-project-Ms-KL

Fly.dev Deployed Project

https://icy-dew-540.fly.dev/projects/

Insomnia

See also:

GET
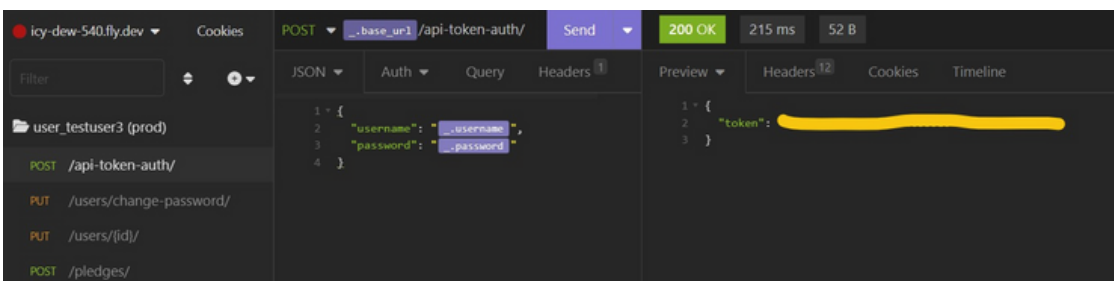
POST

TOKEN

Documentation

Create User

```
curl --request POST \
--url https://icy-dew-540.fly.dev/users/ \
--header 'Content-Type: application/json' \
--data '{
  "username": "<insert_unique_username>",
  "email": "<insert_unique_email>",
  "password":"<insert_password>",
  "bio":"<insert_bio>",
  "avatar":"<insert_url_to_image>"
}'
```

Sign in User

```
curl --request POST \
--url https://icy-dew-540.fly.dev/api-token-auth/ \
--header 'Content-Type: application/json' \
--data '{
  "username": "<insert_unique_username>",
  "password": "<insert_password>"
}'
```
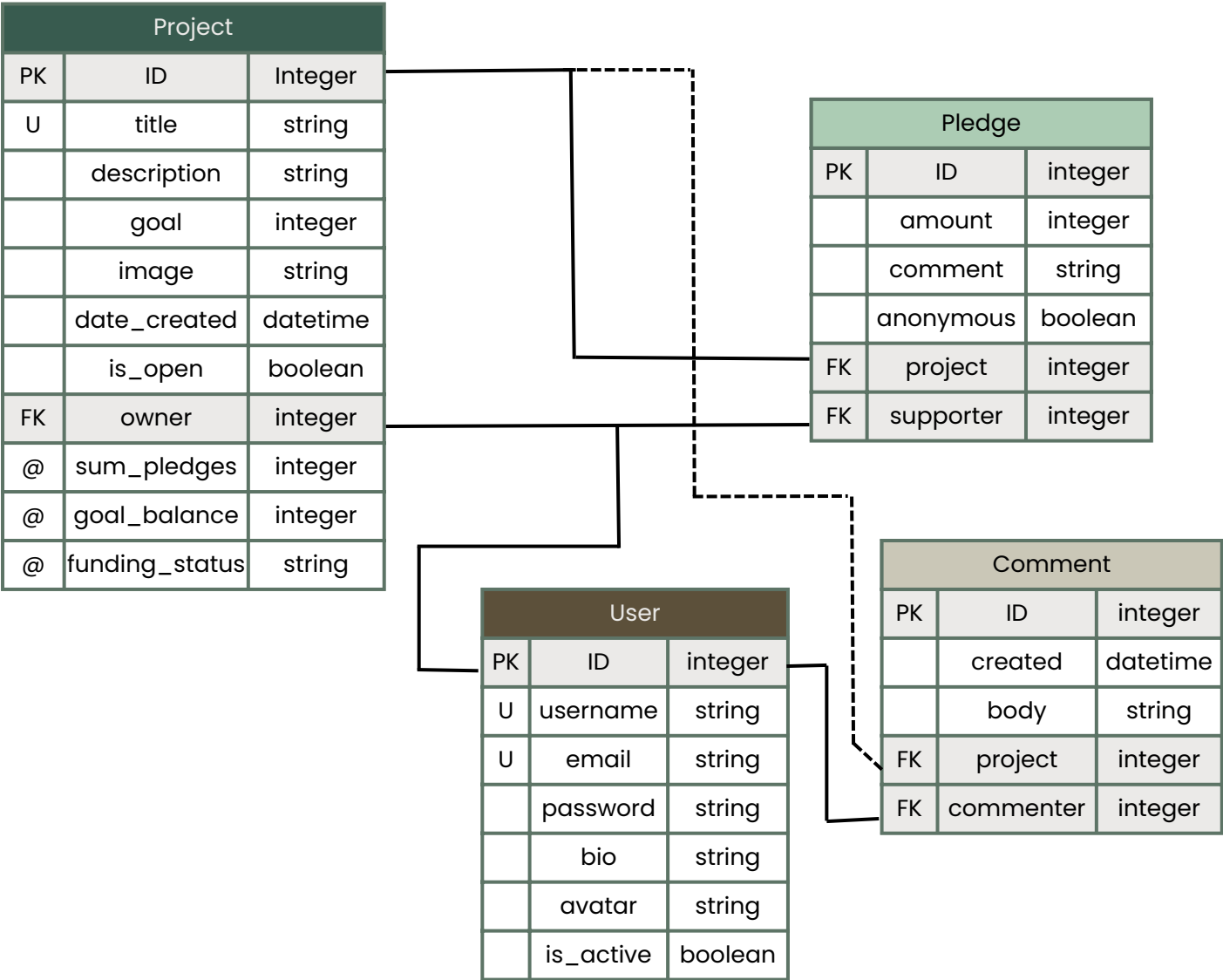
Create Project

```
curl --request POST \
--url https://icy-dew-540.fly.dev/projects/ \
--header 'Content-Type: application/json' \
--data '{
  "title": "<unique_title>",
  "description": "<project_description>",
  "goal": <integer>,
  "image": "<image_url>",
  "is_open": <boolean>,
  "date_created": "<auto_filled as today>"
}'
```

Project Database Schema (Final)

PK: Primary Key | U: Unique | FK: Foreign Key | @: Property

Project

PK	ID	Integer
U	title	string
	description	string
	goal	integer
	image	string
	date_created	datetime
	is_open	boolean
FK	owner	integer
@	sum_pledges	integer
@	goal_balance	integer
@	funding_status	string

Pledge

PK	ID	integer
	amount	integer
	comment	string
	anonymous	boolean
FK	project	integer
FK	supporter	integer

User

PK	ID	integer
U	username	string
U	email	string
	password	string
	bio	string
	avatar	string
	is_active	boolean

Comment

PK	ID	integer
	created	datetime
	body	string
FK	project	integer
FK	commenter	integer

Project API Specification (Final)

HTTP Method	URL	Purpose	Request Body	Successful Response Code	Auth
GET	/projects/	Returns all projects.		200	N/A

POST	/projects/	Create a new project.	{
			"title": "First Project",
			"description": "This is a project",
			"goal": 10,
			"image": "http://image.jpg",
			"is_open": true,
			"date_created": "2023-01-29T09:50:24.154Z"
			}
				201	Logged In

GET	/projects/1/	Returns the project with ID of "1".		200	N/A

PUT	/projects/1/	Updates the project with ID of "1".	{
			"title": "First Project",
			"description": "This is a project",
			"goal": 10,
			"image": "http://image.jpg",
			"is_open": true
			}
				200	Logged In Project Owner

DELETE	/projects/1/	Deletes the project with ID of "1".		200	Logged In Project Owner
POST	/pledges/	Create a new pledge.	{ "amount": 50, "comment": "Good luck!", "anonymous": false, "project": "Bananas" }'	201	Logged In Pledge Owner

GET	/pledges/1/	Get the pledge with ID of "1".	{
			"amount": 10,
			"comment": "I have a dog",
			"anonymous": false,
			"project": "project title",
			"supporter": "username"
			}
				200	N/A

DELETE	/pledges/1/	Deletes the pledge with ID of "1".		200	Logged In Pledge Owner

PUT	/pledges/1/	Updates the pledge with ID of "1"	{
			"comment": "I have a dog",
			"anonymous": false,
			"project": "project title"
			}
				200	Logged In Pledge Owner

GET	/users/	Returns all users.		200	N/A

POST	/users/	Create a new user	{
			"username": "exampleuser",
			"email": "example@example.com",
			"password":"passwordthingy",
			"bio":"Insert bio",
			"avatar":"https://image.jpg"
			}
				201	N/A

PUT	/users/2/	Updates the user with ID of "2".	{
			"username": "exampleuser",
			"email": "example@example.com",
			"is_active":true,
			"bio":"Insert bio",
			"avatar":"https://image.jpg"
			}
				200	Logged In user

POST	/api-token-auth/	creates token	{
			"username": "<insert_unique_username>",
			"password": "<insert_password>"
			}
				201	N/A

Project API Specification (cont...)

HTTP Method	URL	Purpose	Request Body	Successful Response Code	Auth

PUT	/users/change-password/	changes password	{
			"old_password": "thisoldpassword",
			"new_password": "thisnewpassword"
			}
				200	Logged In user

GET	/comments/	Returns all comments.		200	N/A

POST	/comments/	Create a new comment.	{
			"body": "Great Work!",
			"project": 2
			}
				201	Logged In

GET	/comments/1/	Returns the comment with ID of "1".		200	N/A

PUT	/comments/1/	Updates the comment with ID of "1".	{
			"body": "Great Work!",
			"project": 2
			}
				200	Logged In Comment Owner

DELETE	/comments/1/	Deletes the comment with ID of "1".		200	Logged In Comment Owner
GET	projects/? is_open=&owner=	Django-filter returning chosen field values		200	N/A
GET	/pledges/? supporter=&project=	Django-filter returning chosen field values		200	N/A

Additional Features Achieved

Filter Pledges and Projects
Filter pledges by supporter and project. Filter projects by is_open and owner.

User Fields and History
First Name, Last Name, Bio and Avatar added. User Comments, Pledges and Projects history listed in Custom User Detail

Change Password
Change password functionality added

Comments
Comments feature added for users to interact with project

Pledge and Comment History
Pledge and Comment List displayed in Project Detail

Unique field value restrictions
project.title , user.email, user.username restricted; must be unique: throws integrity error with re-entry trigger if not unique

Properties added
sum_pledge, goal_balance & funding_status added in Project Detail