// running tests **(/learn)**
The `POST` request to `/api/issues/{projectname}` will return the created object, and must include all of the submitted fields. Excluded optional fields will be returned as empty strings. Additionally, include `created_on` (date/time), `updated_on` (date/time), `open` (boolean, `true` for open - default value, `false` for closed), and `_id`.
You can send a `GET` request to `/api/issues/{projectname}` for an array of all issues for that specific `projectname`, with all the fields present for each issue.
You can send a `GET` request to `/api/issues/{projectname}` and filter the request by also passing along any field and value as a URL query (ie. `/api/issues/{project}?open=false`). You can pass one or more field/value pairs at once.
You can send a `PUT` request to `/api/issues/{projectname}` with an `_id` and one or more fields to update. On success, the `updated_on` field should be updated, and returned should be `{ result: 'successfully updated', '_id': _id }`.
When the `PUT` request sent to `/api/issues/{projectname}` does not include an `_id`, the return value is `{ error: 'missing _id' }`.
When the `PUT` request sent to `/api/issues/{projectname}` does not include update fields, the return value is `{ error: 'no update field(s) sent', '_id': _id }`. On any other error, the return value is `{ error: 'could not update', '_id': _id }`.
You can send a `DELETE` request to `/api/issues/{projectname}` with an `_id` to delete an issue. If no `_id` is sent, the return value is `{ error: 'missing _id' }`. On success, the return value is `{ result: 'successfully deleted', '_id': _id }`. On failure, the return value is `{ error: 'could not delete', '_id': _id }`. (Test timed out)
All 14 functional tests are complete and passing. (Test timed out)
// tests completed

- ✔ You can provide your own project, not the example URL.
- ✔ You can send a `POST` request to `/api/issues/{projectname}` with form data containing the required fields `issue_title`, `issue_text`, `created_by`, and optionally `assigned_to` and `status_text`.
- ✖ The `POST` request to `/api/issues/{projectname}` will return the created object, and must include all of the submitted fields. Excluded optional fields will be returned as empty strings. Additionally, include `created_on` (date/time), `updated_on` (date/time), `open` (boolean, `true` for open - default value, `false` for closed), and `_id`.
- ✔ If you send a `POST` request to `/api/issues/{projectname}` without the required fields, returned will be the error `{ error: 'required field(s) missing' }`
- ✖ You can send a `GET` request to `/api/issues/{projectname}` for an array of all issues for that specific `projectname`, with all the fields present for each issue.
- ✖ You can send a `GET` request to `/api/issues/{projectname}` and filter the request by also passing along any field and value as a URL query (ie. `/api/issues/{project}?open=false`). You can pass one or more field/value pairs at once.
- ✖ You can send a `PUT` request to `/api/issues/{projectname}` with an `_id` and one or more fields to update. On success, the `updated_on` field should be updated, and returned should be `{ result: 'successfully updated', '_id': _id }`.
- ✖ When the `PUT` request sent to `/api/issues/{projectname}` does not include an `_id`, the return value is `{ error: 'missing _id' }`.
- ✖ When the `PUT` request sent to `/api/issues/{projectname}` does not include update fields, the return value is `{ error: 'no update field(s) sent', '_id': _id }`. On any other error, the return value is `{ error: 'could not update', '_id': _id }`.
- ✖ You can send a `DELETE` request to `/api/issues/{projectname}` with an `_id` to delete an issue. If no `_id` is sent, the return value is `{ error: 'missing _id' }`. On success, the return value is `{ result: 'successfully deleted', '_id': _id }`. On failure, the return value is `{ error: 'could not delete', '_id': _id }`.
- ✖ All 14 functional tests are complete and passing.