Golinks Design Doc Spec

Requirements

- Implement a URL shortener that replaces long URLs with short, easy to remember golinks
- A user will create a link name and link it to the URL they want to shorten
- All shorten URLs will start with ‘go/’ and the link name will follow the slash
- A golink will be distinct for every user. For example, ‘go/git’ will be different for Person A and Person B

- **Note**: This service will only focus on making golinks for individual users and not for an organization.
- The following features will be implemented:
  - User logs in using their Google Account
  - User will be displayed a simple dashboard where they can
    1. View the links they shorten:
       - LinkName - Destination URL - Description - Number of visits
       - User can see how many times they have used the golink
       - User can delete golink
       - User can edit golink
    2. Create a new golink
       - LinkName - Destination URL - Description (Optional)
       - Once user enters the above info, they click the ‘Create button’ and can view the new golink in their dashboard

Example

- Golink: ‘go/git’
- Destination url: ‘https://github.com/ocastroa’
- Description: ‘My GitHub repository home page’

API Design

- Identify activities
  - User logs into their account with Google sign-in
  - User can view their basic info (name, email)
  - User can delete their account
  - User can view their dashboard that shows the golinks they have created
  - User can click a golink in their dashboard and will be directed to the destination URL
  - User can enter a golink in the browser search bar and will be redirected to the destination URL
  - User can delete a golink from their dashboard
  - User can edit the destination URL in their dashboard
  - User can edit the description info in their dashboard
  - User can create a new golink
  - User can filter data for specific link names or destination URLs (done on the client side)

- Break into steps
  - Auth Resource - Validate Google ID token and add user to database if user is new, generate sessions id and send back to the client as a cookie, log user out
  - Users Resource - View user, delete user
  - Links Resource – List all golinks, create a new golink, edit golinks information, delete a golink
  - Routings Resource – Redirect to destination URL

- Auth Middleware – Checks if sessions token is valid for the user and gets user's email

- Status codes
  - Successful Response
    - 200 – OK => Request succeeded
    - 201 – Created => New resource has been created
    - 204 – No Content => No content to send back. Use for DELETE requests
  - Redirection Messages
    - 302 – Found => Use to redirect user within the site or to the destination URL
  - Client Error Messages
    - 400 – Bad Request => Invalid syntax in payload or URI
    - 401 – Unauthorized => User is not logged in and does not have a sessions cookie. User cannot access endpoints
    - 404 – Not Found => Requested resource was not found
    - 409 – Conflict => User is trying to add a link name that already exists. Link names must be unique for each user
  - Server Error Responses
    - 500 – Internal server error

- API Definition
  - Payload (for more details, look at the end of this doc):
    - {} => object
    - [{}] => array of objects
  - Auth Resource

Activity	Verb	/Noun mapping	Request Payload	Response Payload	Success code	Error code
Log user out and remove session token	GET	/v1/auth/logout	Nothing	Nothing	302 (redirect to login page)	302 (redirect to login page)
Validate Google token and add new users to db	POST	/v1/auth	{}	Nothing	302 (redirect user to their dashboard)	302 (redirect to login page)

- Users Resource
  - User can view their information in their profile, as well as delete their account
  - User's email, which is stored in the sessions db, will be used to differentiate each user

Activity	Verb	/Noun mapping	Request Payload	Response Payload	Success code	Error code
View user	GET	/v1/users	Nothing	{}	200	404 or 401
Delete User	DELETE	/v1/users	Nothing	Nothing	302 (redirect to login page)	404 or 401

- Links Resource
  - User can only edit destination URL and description, not link name

Activity	Verb	/Noun mapping	Request Payload	Response Payload	Success code	Error code
List all Links	GET	/v1/links	Nothing	[{}]	200	401
Create a new Link	POST	/v1/links	{}	{}	201	400 or 409 or 401
Edit a Link	PUT	/v1/links/:link_name	{}	{}	200	400 or 302 or 401
Delete a Link	DELETE	/v1/link/:link_name	Nothing	Nothing	204	401

- Routings Resource - Redirect to destination url

Activity	Verb	/Noun mapping	Request Payload	Response Payload	Success code	Error code
Redirect to destination URL	GET	/v1/routings/:link_name	Nothing	Nothing	302	302 (redirect to dashboard)

Database Design

- Users
  - Userid
  - First name
  - Last name
  - Email
  - Created on

PK (INTEGER)	user_id
VARCHAR(100)	first_name
VARCHAR(100)	last_name
VARCHAR(100), Unique	email
TIMESTAMP	created_on

- Links
  - LinkId
  - LinkName
  - Destination URL
  - Description (Not required)
  - Visits Count
  - Email (who created it)

PK (INTEGER)	link_id
VARCHAR(100)	link_name
VARCHAR(2000)	destination_url
TEXT	description
INTEGER, DEFAULT(0)	visits_count
VARCHAR(100)	email

- Sessions
  - id
  - user_email
  - session_id
  - create_date

PK (INTEGER)	id
VARCHAR(100)	user_email
VARCHAR(100)	session_id
TIMESTAMP	created_date

Payload

Auth Resource:

1) Revoke session token when user signs out

Request: Nothing
Response: Nothing

2) Validate Google id_token and add new users to db. Send back session id_token

Request: Get from OAuth 2.0 Playground

```
{
    access_token: string
}
```

Response: Nothing

Users Resource:

1) View User:

Request: Nothing

Response:

```
{
    user_id: (integer) user id,
    first_name: (string) user's first name,
    last_name: (string) user's last name,
    email: (string) user's google email address,
    created_on: (date) timestamp that user was added to db
}
```

Example response:

```
{
    user_id: 1
    first_name: 'Oscar',
    last_name: 'Castro',
    email: ‘oscarcastro@gmail.com',
    created_on: '2020-01-26 23:38:39'
}
```

2) Delete a user:

Request: Nothing

Response: Nothing

Links Resource:

1) List all links:

Request: Nothing

Response:

```
[
    {
        link_id: (integer) id of link,
        link_name: (string) name of short link,
        destination_url: (string) url of destination link,
        description: (string) short description of link,
        visits_count: (integer) number of times user visited link,
        email: (integer) email of user that created golink
    },
    ...
]
```

Example response:

```
[
    {
        link_id: 1,
        link_name: 'git',
        destination_url: 'https://github.com/ocastroa',
        description: 'My GitHub repo',
        visits_count: 2,
        email: ‘oscarcastro@gmail.com’
    }
]
```

2) Create a new link

Request:

{

link_name: (string) name of short link,

destination_url: (string) url of destination link,

description: (string) short description of link (optional)

}

Response:

{

link_id: (integer) id of link,

link_name: (string) name of short link,

destination_url: (string) url of destination link,

description: (string) short description of link,

visits_count: (integer) number of times user visited link,

email: (integer) email of user that created golink

}

3) Edit a link

Path Param: 'link_name'

Request:

{

destination_url: (string) url of destination link,

description: (string) short description of link (optional)

}

Response:

{

link_id: (integer) id of link,

link_name: (string) name of short link,

destination_url: (string) url of destination link,

description: (string) short description of link,

visits_count: (integer) number of times user visited link,

email: (integer) email of user that created golink

}

4) Delete a link:

Path Param: 'link_name'

Request: Nothing

Response: Nothing

Redirect Resource

1) Redirect user to destination url

Path Param: 'link_name'

Request: Nothing

Response: Nothing