Logisteps Controllers

# Overview

This document provides design details regarding the web application’s controllers. Each controller, known in Django as a View, handles requests by querying and processing data. Once the request has been fulfilled, the controller sends a response back to the requesting client. Consequently, there a controller must be defined for each valid URL – whether that is for retrieving web pages or manipulating/accessing data models. Since the Logisteps web application must handle requests from the mobile app client (for getting/creating user data and posting step data) in addition to serving web pages to users wishing to view their data, there are two logically separated categories of controllers – web page views and rest framework views. This document will document design for all controllers necessary to fulfill system requirements.

The controllers sit facilitate communication between external agents, templates/views, and data models. More information regarding controllers (Django Views) can be found on their [website](https://docs.djangoproject.com/en/2.1/topics/class-based-views/).

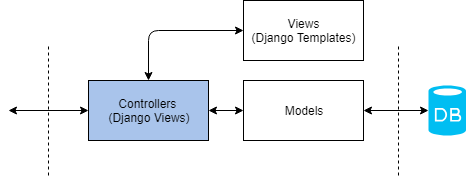


Figure - Controllers handle requests and return responses.

# Web Page Controllers

This set of vies provides should provide the functionality for displaying different web pages to a user. When users navigate to a URL in their browser, the controllers defined in this section will handle the GET requests and respond with the appropriate rendered HTML file.

For example, when a user navigates to /logisteps/login.html, the Login controller should handle the GET request for serving the web page, as well as handle the POST request that occurs when the user submits the form data.

## Controllers

### Register

#### Purpose

Provide an interface for users to register new accounts if they have never logged into the system before. This controller should serve an HTML page for submitting form data if a GET request is made and should handle data submitted from the client in a POST request.

#### URL

/logisteps/register/

#### Authentication

Login not required

#### Supported HTTP Methods

##### GET

* Action: Serve the registration template to the client. This should include a form for posting user data.
* Parameters:

None

##### POST

* Action: Handle form data from client and create a new LogistepsUser model.
* Parameters:

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Type | Constraints | Description |
| username | string | unique | Username for new account |
| email | string |  | Email for user. Should be validated client side |
| first\_name | string | max\_length=50 | User’s first name |
| last\_name | string | max\_length=50 | User’s last name |
| password | string |  | User’s password. Password validation should be performed client side. |

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

### Complete Profile

#### Purpose

After completing registration on the registration page, a Django User model will be saved in the database, but a LogistepsUser model cannot be created because there is not enough information to create the model yet. This controller provides an interface for completing a user profile.

#### URL

/logisteps/profile/complete/

#### Authentication

Login required.

#### Supported HTTP Methods

##### GET

* Action: Serve the template for completing a user profile. This should include a form for posting user data.
* Parameters:

None

##### POST

* Action: Handle the form data from the client and create a new LogistepsUser model.
* Parameters:

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Type | Constraints | Description |
| left\_shoe | decimal | 4 <= value <= 16  max\_digits=3  max\_decimal=1 | Size of the user’s left foot/insole. Validation should be performed client side. |
| right\_shoe | decimal | 4 <= value <= 16  max\_digits=3  max\_decimal=1 | Size of the user’s right foot/insole. Validation should be performed client side. |
| height\_feet | integer | value > 0 | User’s height in feet. |
| height\_inches | integer | value > 0 | User’s height in inches. |
| weight | integer | value > 0 | User’s weight in lbs. |
| step\_goal | integer | value >= 0 | User’s daily step goal. |

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

### Login

#### Purpose

This controller provides a login page to a client and handles form data from the login screen when form data is posted. Upon successful login, this controller should automatically redirect a user to the landing page of the Logisteps web application.

*Note:* Django provides a default controller which can handle this functionality. The only thing that needs to be done is implementing the login.html template.

#### URL

/accounts/login/

#### Authentication

Login not required

#### Supported HTTP Methods

##### GET

* Action: Serve the login.html template for authenticating a user. Should provide a form for posting required data.
* Parameters

None

##### POST

* Action: Handle for data and check if user credentials are valid. Redirect to landing page if successful; show error message if unsuccessful.
* Parameters

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Type | Constraints | Description |
| username | string |  | User’s username for their account |
| password | string |  | User’s password associated with their account. |

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

### Profile

#### Purpose

This controller should provide a means for users to view and update their profile information. In particular, the controller should serve a form for updating LogistepsUser fields, and then handle any POST request when the form is submitted by the user. Upon successful processing of the POST, users will see their updated information immediately.

#### URL

/logisteps/profile/

#### Authentication

Login required

#### Supported HTTP Methods

##### GET

* Action: Serve profile.html to the user with a form for updating their information.
* Parameters:

None

##### POST

* Action: Handle user profile data submitted from the form and update the user’s LogistepUser model. Should reload the page after success, and display error if there was an error.
* Parameters:

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Type | Constraints | Description |
| left\_shoe | decimal | 4 <= value <= 16  max\_digits=3  max\_decimal=1 | Size of the user’s left foot/insole. Validation should be performed clientside. |
| right\_shoe | decimal | 4 <= value <= 16  max\_digits=3  max\_decimal=1 | Size of the user’s right foot/insole. Validation should be performed clientside. |
| height\_feet | integer | value > 0 | User’s height in feet. |
| height\_inches | integer | value > 0 | User’s height in inches. |
| weight | integer | value > 0 | User’s weight in lbs. |
| step\_goal | integer | value >= 0 | User’s daily step goal. |

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

### Logout

#### Purpose

This controller should provide a logged\_out.html template to the client indicating that the user has been successfully logged out, with a link for logging back into the application.

*Note:* Django provides a default logout controller. A template will be provided to the default controller.

#### URL

/logisteps/logout/

#### Authentication

Login not required, but is a often a precursor to logout.

#### Supported HTTP Methods

##### GET

* Action: Should return a HTML document to the client indicating that they have successfully logged out.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

### Forgot Password

#### Purpose

This controller should allow a user to reset their password by serving a form to the client and handling subsequent POST requests. This controller will require users to reference resources outside of the webpage to verify identity.

#### URL

/accounts/reset/

#### Authentication

Login is not required since the user has forgotten their password, but they will be required to verify their identity using their email address that they provided when creating a profile.

#### Supported HTTP Methods

##### GET

* Action: Serves a webpage with a form for resetting a password.
* Parameters:  
  None

##### POST

* Action: Should process request and change the user’s password if user verification was successful. If verification fails, the user should be notified and prompted to try again.
* Parameters:

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Type | Constraints | Description |
| username | string |  | User’s username that they user to login |
| email | string |  | User’s email that they used to register |
| verification\_code | string |  | Code sent to user’s email to verify identity |

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

### Recent

#### Purpose

This controller should handle returning the HTML template that displays a user’s step summary from the current day and the previous page. Due to the nature of Django databinding and JavaScript graphing libraries, the actual data for the graphics will not be queried and processed in this controller. Rather, this controller will respond with an HTML page that has embedded JavaScript which makes calls to the rest API for populating graph data.

#### URL

/logisteps/recent/

#### Authorization

Login Required

#### Supported HTTP Methods

##### GET

* Action: Return a rendered HTML page for displaying the user’s step summary for the previous two days, as defined by the system requirements.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

### Steps Over Time

#### Purpose

This controller should handle returning the HTML template that displays a user’s steps over time for the current week. Due to the nature of Django databinding and JavaScript graphing libraries, the actual data for the graphics will not be queried and processed in this controller. Rather, this controller will respond with an HTML page that has embedded JavaScript which makes calls to the rest API. The returned page will have an adjustable filter but changing the filter will be handled by the embedded JavaScript/REST framework.

#### URL

/logisteps/steps\_over\_time/

#### Authorization

Login Required

#### Supported HTTP Methods

##### GET

* Action: Return a rendered HTML page for displaying the user’s steps over a week, as defined by the system requirements.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

### Steps by Weekday

#### Purpose

This controller should handle returning the HTML template that displays a cumulative steps per weekday graphic. Due to the nature of Django databinding and JavaScript graphing libraries, the actual data for the graphics will not be queried and processed in this controller. Rather, this controller will respond with an HTML page that has embedded JavaScript which makes calls to the rest API to populate the graph data.

#### URL

/logisteps/steps\_by\_weekday/

#### Authorization

Login Required

#### Supported HTTP Methods

##### GET

* Action: Return a rendered HTML page for displaying the user’s steps per weekday, as defined by the system requirements.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

### Activity by Week

#### Purpose

This controller should handle returning the HTML template that displays a user’s activity time vs inactive time. Due to the nature of Django databinding and JavaScript graphing libraries, the actual data for the graphics will not be queried and processed in this controller. Rather, this controller will respond with an HTML page that has embedded JavaScript which makes calls to the rest API to populate the graph data. The returned page will have an adjustable filter but changing the filter will be handled by the embedded JavaScript/REST framework.

#### URL

/logisteps/active\_time/

#### Authorization

Login Required

#### Supported HTTP Methods

##### GET

* Action: Return a rendered HTML page for displaying the active time vs their inactive time, as defined by the system requirements.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

### Pressure

#### Purpose

This controller should handle returning the HTML template that displays a user’s foot pressure over time. Due to the nature of Django databinding and JavaScript graphing libraries, the actual data for the graphics will not be queried and processed in this controller. Rather, this controller will respond with an HTML page that has embedded JavaScript which makes calls to the rest API to populate the graph data. The returned page will have an adjustable filter but changing the filter will be handled by the embedded JavaScript/REST framework.

#### URL

/logisteps/pressure/

#### Authorization

Login Required

#### Supported HTTP Methods

##### GET

* Action: Return a rendered HTML page for displaying the user’s pressure over time, as defined by the system requirements.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

### Map

#### Purpose

This controller should handle returning the HTML template that displays a step location for the past day. Due to the nature of Django databinding and JavaScript graphing libraries, the actual data for the graphics will not be queried and processed in this controller. Rather, this controller will respond with an HTML page that has embedded JavaScript which makes calls to the rest API to populate the graph data. The returned page will have an adjustable filter but changing the filter will be handled by the embedded JavaScript/REST framework.

#### URL

/logisteps/map/

#### Authorization

Login Required

#### Supported HTTP Methods

##### GET

* Action: Return a rendered HTML page for displaying the user’s step location over the past day, as defined by the system requirements.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

# REST API Controllers

The web page controllers’ primary responsibilities were rendering and returning interactive HTML documents to the client as a way for a user to view and manipulate data. For controllers that return HTML documents with forms, they also implement functionality for processing data in POST requests which create/update Logistep models. REST API Controllers differ greatly from the web page controllers. Rather than serving web pages and handling responses, the primary responsibility for the REST API controllers is to return JSON documents representing Logisteps models, statistics, and queries. In addition to returning JSON documents, the other primary responsibility for the REST API controllers is to manipulate and create models by interpreting JSON documents sent to the web server via POST, PUT, and DELETE HTTP methods.

The REST API Controllers expose data that would normally not be exposed by Django and are meant to be used by the mobile app and JavaScript embedded in the Django HTML templates. Most REST API controllers require basic authentication.

The REST API is documented in detail in the API documentation, so this section will not go into as much detail as the Web Page Controllers section. Developers wishing to use the API should consult the API documentation found at the following link.

REST API documentation: <https://github.com/SeniorDesignTeamOmicron/Documentation/tree/master/Project%20Design/web%20application/API>