NeuroCage Test Documentation

Release 1.0

Ali ESMAEILI

May 12, 2023

Contents

1 Project Introduction

a. Contents

b. Views

c. LICENSE

2 Project Structure

a. Code Repository usage

b. Services

c. Database Design

3 Project Usage

a. Clone Repository

b. Setup Environments Variables

c. Docker Compose Up

d. Deployment Facts

e. Tests

CHAPTER 1

Project Introduction

a Contents

NeuroCage test is a Django project designed to showcase my skills as a developer and apply for a position at Neurocage Company.

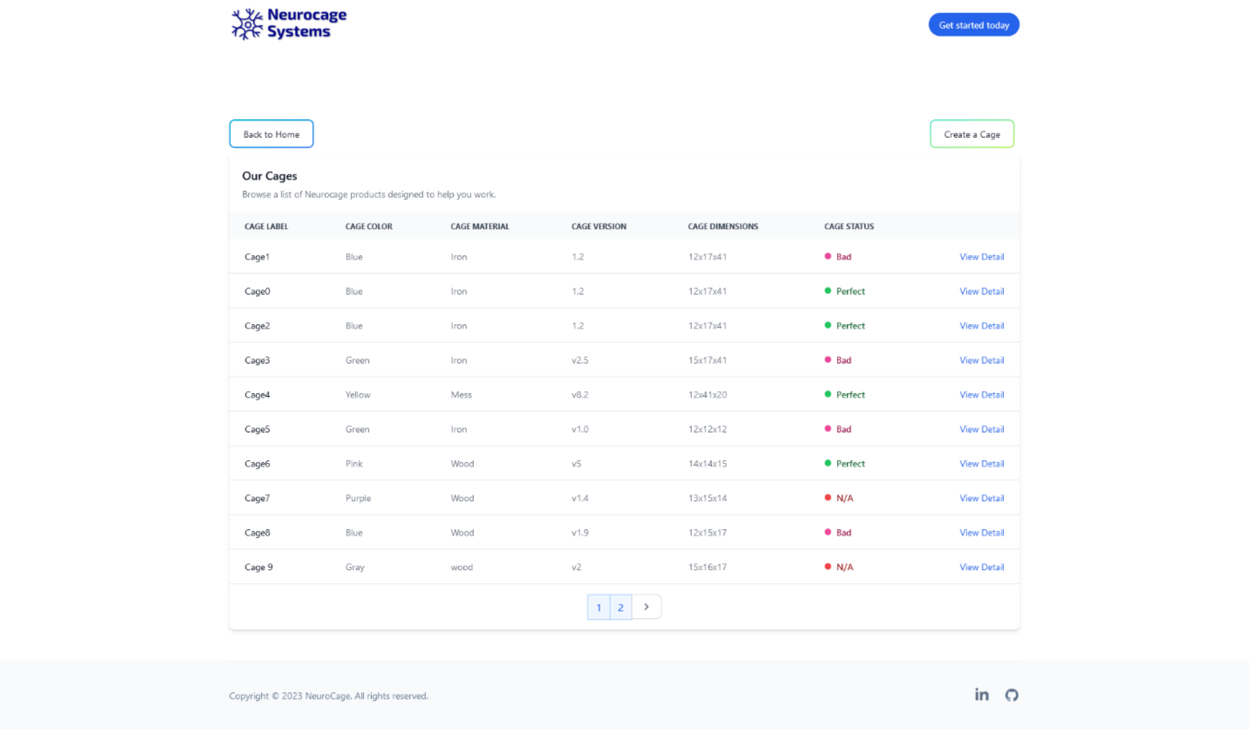
NeuroCage Documents summary: **(PDF Version)**

The project aims to design a web application that allows a user to manage smart cages developed by Neurocage. The web application should allow the user to view a list of cages, create a new cage, and view the details of a cage on a separate page. Each cage should have an ID and label field, and the developer can add any other necessary fields. Additionally, the application should save sensor data from the cages by calling the endpoint of a separate project that simulates a sensor. The sensor fails to respond 25% of the time, but the remaining 75% of the time, it sends a value representing the health of the cage. The web application should call the sensor endpoint every 10 minutes to save the health values to a time series. The application should also create a table to record the time it takes to get a response from the sensor and whether the sensor succeeded or failed.

b Views

Cage List:

- Add Pagination
- Should contain a link to the detail page of each cage
- Should show the latest health status for each cage



Our Cages

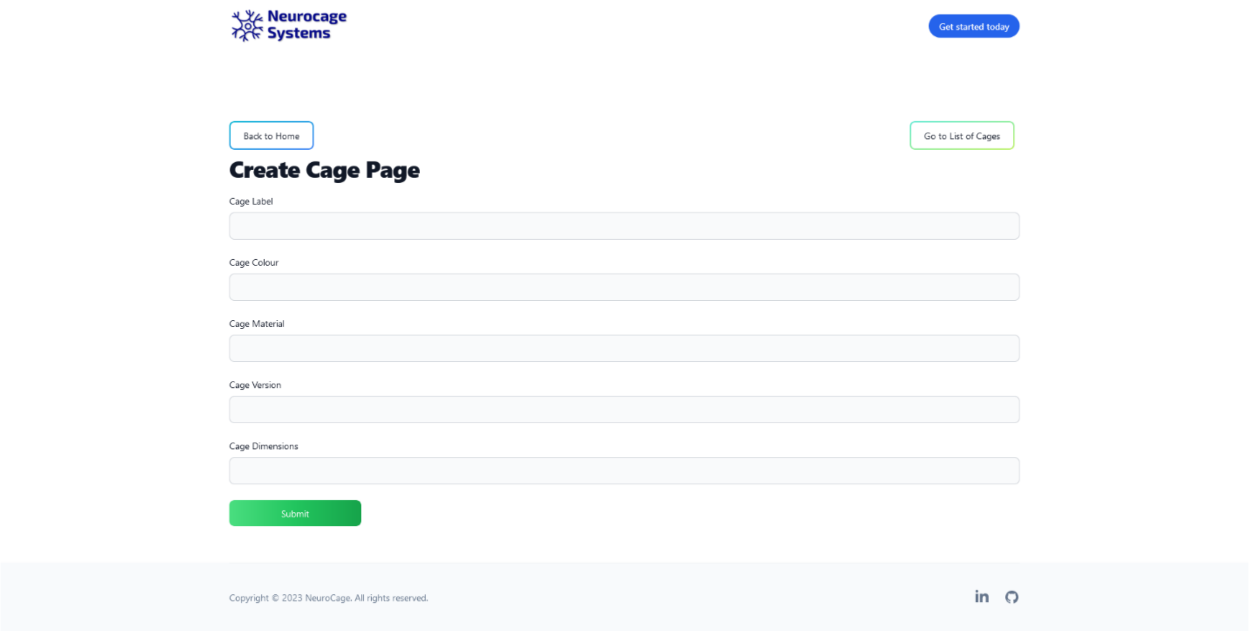
Browse a list of Neurocage products designed to help you work.

CAGE LABEL	CAGE COLOR	CAGE MATERIAL	CAGE VERSION	CAGE DIMENSIONS	CAGE STATUS	
Cage1	Blue	Iron	1.2	12x17x41	● Bad	View Detail
Cage0	Blue	Iron	1.2	12x17x41	● Perfect	View Detail
Cage2	Blue	Iron	1.2	12x17x41	● Perfect	View Detail
Cage3	Green	Iron	v2.5	15x17x41	● Bad	View Detail
Cage4	Yellow	Mess	v8.2	12x41x20	● Perfect	View Detail
Cage5	Green	Iron	v1.0	12x12x12	● Bad	View Detail
Cage6	Pink	Wood	v5	14x14x15	● Perfect	View Detail
Cage7	Purple	Wood	v1.4	13x15x14	● N/A	View Detail
Cage8	Blue	Wood	v1.9	12x15x17	● Bad	View Detail
Cage 9	Gray	wood	v2	15x16x17	● N/A	View Detail

Copyright © 2023 NeuroCage. All rights reserved.

New Cage:

- Nothing fancy, just create the cage!

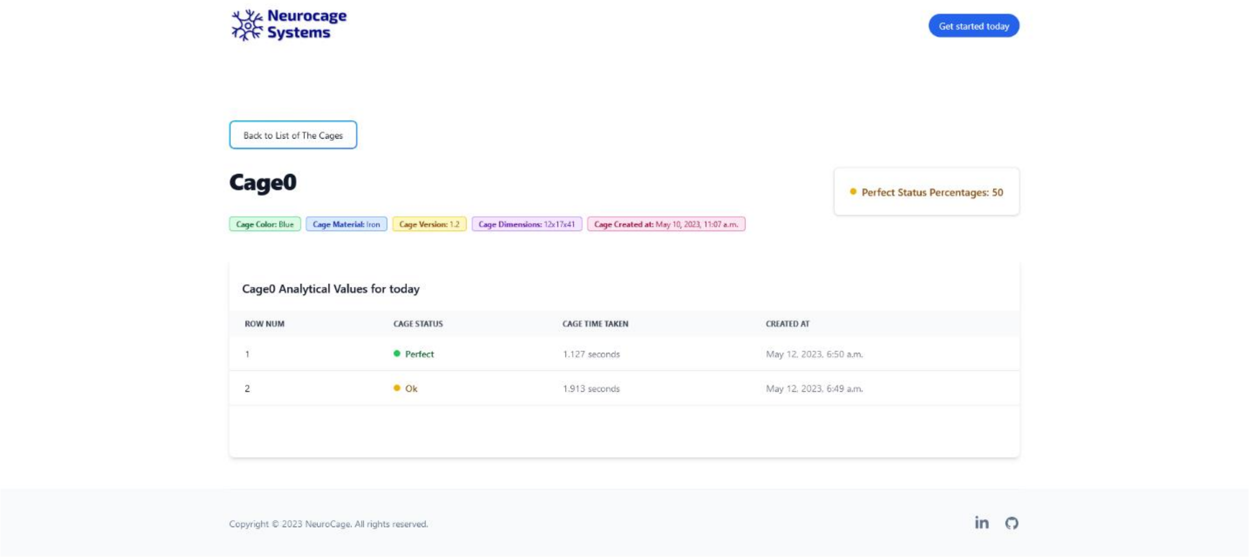


Create Cage Page

Copyright © 2023 NeuroCage. All rights reserved.

Detail Page:

- Information about the cage
- Health values for today
- The functionality of the sensor for today



Cage0

Cage Color: Blue | Cage Material: Iron | Cage Version: 1.2 | Cage Dimensions: 12x17x41 | Cage Created at: May 10, 2023, 11:07 a.m.

● Perfect Status Percentages: 50

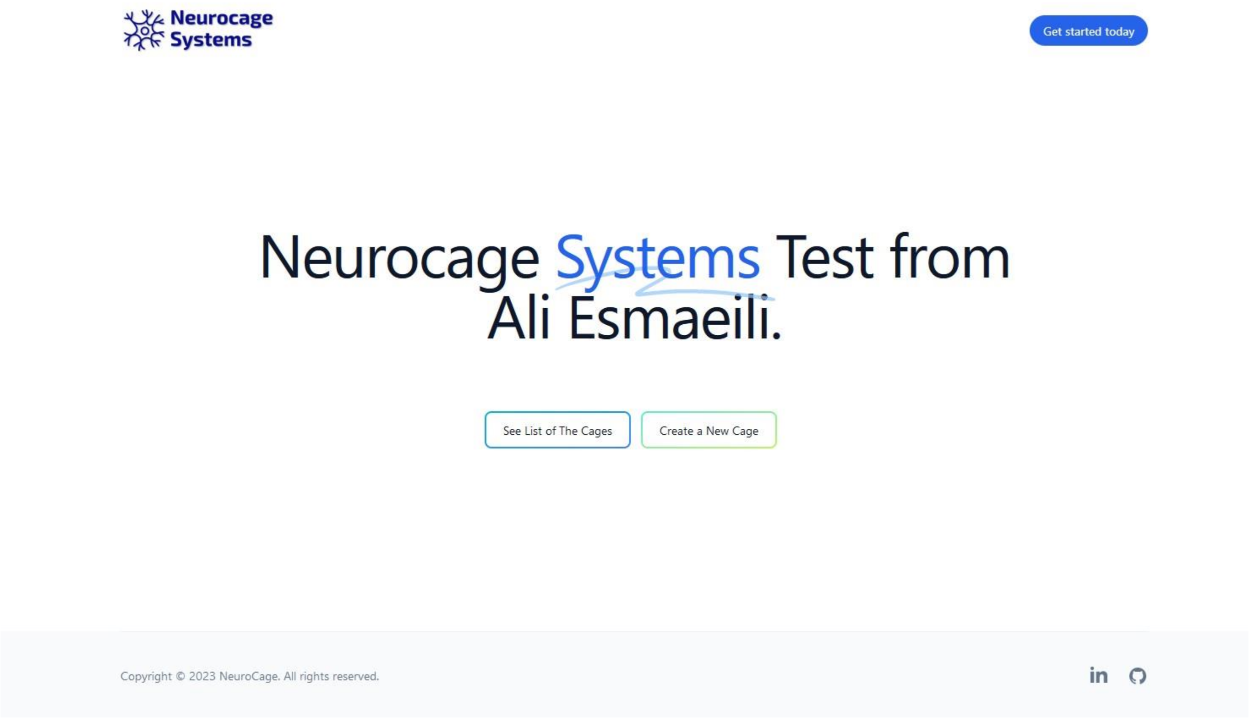
Cage0 Analytical Values for today

ROW NUM	CAGE STATUS	CAGE TIME TAKEN	CREATED AT
1	● Perfect	1.127 seconds	May 12, 2023, 6:50 a.m.
2	● Ok	1.913 seconds	May 12, 2023, 6:49 a.m.

Copyright © 2023 NeuroCage. All rights reserved.

Home Page:

- Just a page that users can select create or list cage button to redirect to these pages



Neurocage Systems Test from Ali Esmaeili.

Copyright © 2023 NeuroCage. All rights reserved.

c LICENSE

This project is licensed under the GNU General Public License v3.0 - see the LICENSE file for details.

CHAPTER 2

Project Structure

a Code Repository usage

A Github Repository. Simply Clone it and push your changes directly.

In Development states you should use dev branch and if you have a new feature, It’s really recommended that you create a new branch for develop the feature until the development of the feature has done.

URL of Github Repo:

https://github.com/realxoman/neurocage_test/

b Services

The following services are included in the project:

- web: The Django web application that serves the main functionality of the project. It runs on port 8080 and communicates with the database, the message broker, and the Celery worker.
- db: The PostgreSQL database that stores the data of the application. It is configured with environment variables from the .env file and persists its data in a Docker volume.
- nginx: The Nginx server that acts as a reverse proxy for the web application. It serves static and media files and logs its activity to a local directory.
- rabbit: The RabbitMQ message broker that handles the communication between the web application and the Celery worker. It runs on its default ports 5672 and 15672.
- celery: The Celery worker that executes background tasks asynchronously. It is configured to use the RabbitMQ broker and runs with a concurrency of 10 processes.

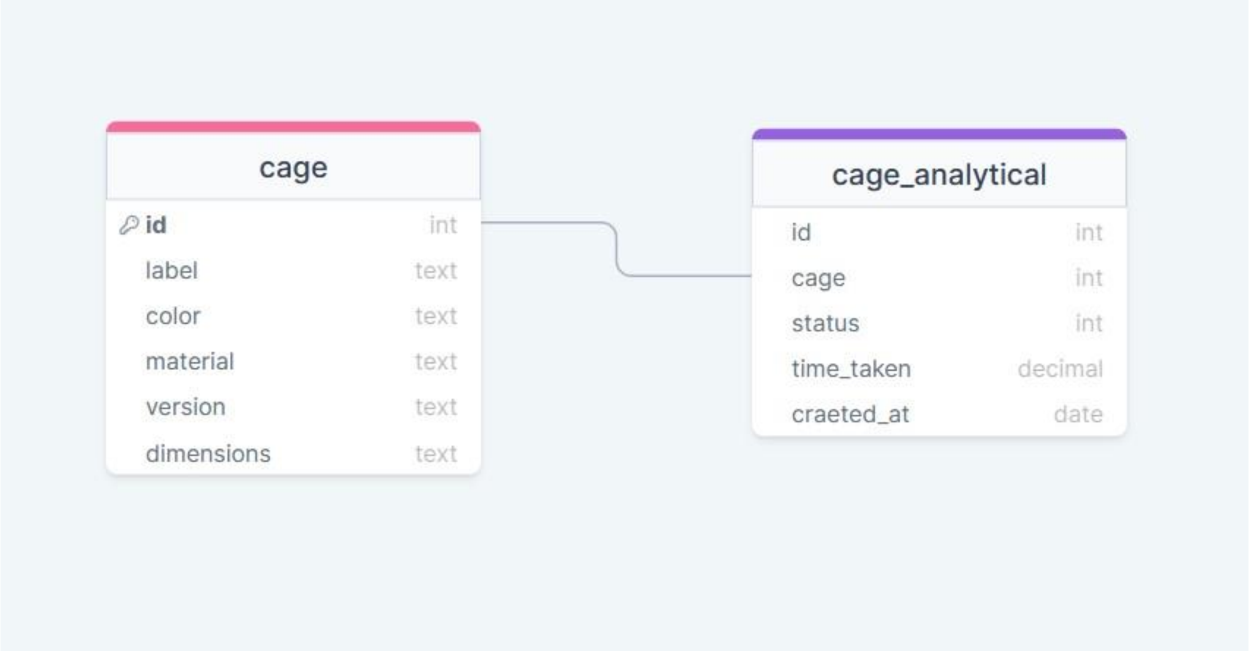
c Database Design

1 – Cage

A.	**Id**	[int, Autoincrement, Primary Key]
B.	**Label**	[varchar, max_length=256, null=True]
C.	**Material**	[varchar, max_length=50, null=True]
D.	**Version**	[varchar, max_length=10, null=True]
E.	**Dimensions**	[varchar, max_length=50, null=True]
F.	**Created_at**	[DateTime, Default=timezone.now()]

2 – Cage Analytical

A.	**Id**	[int, Autoincrement, Primary Key]
B.	**Cage**	[Fk, Cage]
C.	**Status**	[Uint]
D.	Time_taken	[Decimal]
E.	Created_at	[DateTime, Default=timezone.now()]



cage	
id	int
label	text
color	text
material	text
version	text
dimensions	text

cage_analytical	
id	int
cage	int
status	int
time_taken	decimal
craeted_at	date

CHAPTER 3

Project Usage

a Clone Repository

Clone this repository to your machine:

```
git clone https://github.com/realxoman/neurocage_test.git
```

b Setup Environments Variables

Navigate to the project directory:

```
cd neurocage_test
```

Copy the example environment file and adjust the settings if necessary:

```
cp env.dev .env
```

c Docker Compose Up

Build and start the containers in detached mode:

```
docker-compose up --build -d
```

Open your web browser and visit http://localhost to see the web application in action.

d Deployment Facts

It’s better to configurate the Nginx and Database in the root of the server to manage better. So If you want to deploy this project in best practice, I recommended to you that use the docker-compose-stg.yml file.

e Tests

The tests in this documentation cover various aspects of the application, including Flask requests, Django forms, models, URLs, and views. The test_flask_requests test checks the response and status code of all Flask requests made in the application. The test_forms test checks the validity of all Django forms used in the application. The test_models test ensures that all Django models used in the application are properly constructed and perform their intended functions. The test_urls test checks the validity and functionality of all URLs used in the application, while the test_views test checks the response and functionality of all Django views used in the application.

To run all of the tests, you can use this command:

```
python manage.py tests
```

End.