1 Installation 3
  1.1 Set up the database ...................................................... 3
  1.2 View a demo image ..................................................... 3
  1.3 Add a tiled image ....................................................... 3

2 Contributing to VirtualMicroscope 5
  2.1 Contributing ............................................................. 5
  2.2 Contributing Documentation ......................................... 5

3 Future development 7
  3.1 System architecture .................................................. 7
  3.2 Docker ................................................................. 7
  3.3 Leaflet.js as an alternative to Google Maps ....................... 7

4 Indices and tables 9
VirtualMicroscope is a Python/Django application to view, navigate and annotate very high-resolution images.

It was created at New York University School of Medicine for viewing medical slides for teaching and research purposes, but in fact is suitable for exploring any high-resolution images - from astronomy, museum collections, specialised photographic applications and so on.

The application was written by William Holloway and Marc Triola of NYU, where it has been used sucessfully and written up in Enhanced virtual microscopy for collaborative education.

In January 2015 VirtualMicroscope was selected by a team at NHS HackDay in Cardiff, to help make it easier to deploy and work with.

This version of VirtualMicroscope is a fork of the original version hosted by NYU on Google Code.

Contents:
VirtualMicroscope is hosted on GitHub.

To install:

```bash
git clone -b hackday git@github.com:evildmp/VirtualMicroscope.git  # use the hackday branch
cd VirtualMicroscope
virtualenv env  # create a virtual environment
source env/bin/activate  # activate it
pip install -e .  # run setup.py and install required components
```

### 1.1 Set up the database

This repository includes demo images and a database in JSON format, so we’ll use those to get started.

Create your local database, and import the serialised JSON data:

```bash
python manage.py syncdb --noinput  # noinput will prevent it asking for superuser details
python manage.py loaddata example_database.json
python manage.py runserver
```

Assuming you’re running on localhost on port 8000, visit [http://localhost:8000/admin/](http://localhost:8000/admin/) and login:

```
{'username': '*vm*'}
{'password': '*vm*'}
```

### 1.2 View a demo image


And that’s it - now you can zoom in and out on people at the HackDay event; our team’s the one at the bottom right.

### 1.3 Add a tiled image

We’ve included a second tiled image, but you need to add this yourself. That image has been tiled for you already, and is to be found in virtualmicroscope/static/auraya.

In the Django admin, create a new Slide:
• URL to slide directory: \url{http://localhost:8000/static/auraya}
• Label: whatever you like
• Maximum Google Zoom Level: 4

Save the Slide.
Add a new \textit{Collection}, and make sure you are in the \textit{Authors}.
Add a \textit{Collection Slide}, that refers to your news Slide and your new Collection; fill in the other fields as you like.
Visit \url{http://localhost:8000/virtualmicroscope}.
Choose your Collection from the \textit{Collections} menu, and choose your slide below.
And that’s it - now you can zoom in and out on Auraya the dog.
Like every open-source project, VirtualMicroscope is always looking for motivated individuals to contribute to its source code.

## 2.1 Contributing

People interested in contributing to VirtualMicroscope can visit #virtualmicroscope on the freenode IRC network for help and to discuss the development.

VirtualMicroscope is hosted on GitHub.

The best method to contribute back is to create an account there, then fork the project. You can use this fork as if it was your own project, and should push your changes to it.

When you feel your code is good enough for inclusion, “send us a pull request”, by using the nice GitHub web interface.

## 2.2 Contributing Documentation

Documentation should be:

- written using valid Sphinx/restructuredText syntax (see below for specifics) and the file extension should be .rst
- written in English (we have standardised on British spellings)
- accessible - you should assume the reader to be moderately familiar with Python and Django, but not anything else. Link to documentation of libraries you use, for example, even if they are “obvious” to you
- wrapped at 100 characters per line

### 2.2.1 Documentation markup

**Sections**

We use Python documentation conventions for section marking:

- # with overline, for parts
- * with overline, for chapters
• =, for sections
• –, for subsections
• ^, for subsubsections
• "", for paragraphs

Inline markup

• use backticks - `settings.py` - for:
  – literals
  – filenames
  – names of fields and other items in the Admin interface:

• use emphasis - *Home* around:
  – the names of available options in the Admin
  – values in or of fields

• use strong emphasis - **Add page** around:
  – buttons that perform an action

References

Use absolute links to other documentation pages - :doc:`/installation` - rather than relative links - :doc:`../installation`. This makes it easier to run search-and-replaces when items are moved in the structure.
A few ideas for future development are listed below.

### 3.1 System architecture

In order to cover the largest number of deployment scenarios, a slightly refactored architecture for our system is proposed here.

Without losing much generality, we should be able to assume that image files are already stored on the local filesystem and available via a local path. Potentially, they are very large images and uploading them through a web-browser might not make much sense anyway.

With this assumption in mind, the following workflow is then possible.

Additional feature: Django will allow the admin user to browse the local image folder and mark the ones that are meant to be served through the system.

Additional feature: once the selection is confirmed, Django will start processing the images and creating the tiles as a background task (Celery may come in here). Once the processing is done for a particular image, this is marked as ready on the Django admin and will be available for viewing to the final user.

The other parts of the software - most notably the front-end part that allows images to be displayed - would remain as currently implemented.

### 3.2 Docker

Given the above architecture, the Docker Engine could be used to create a dedicated image and containerize our whole application.

The image folder could be made available to Docker as a volume via the `-v` option.

### 3.3 Leaflet.js as an alternative to Google Maps

Leaflet is an open source JavaScript library to build web mapping applications, akin to the Google Maps API currently in use in the project.

Leaflet could be taken into account as a replacement to Google Maps since the latter is a proprietary solution.
The integration of Leaflet in VirtualMicroscope should be feasible without major issues. A few references are listed below.

- http://kempe.net/blog/2014/06/14/leaflet-pan-zoom-image.html
- http://hugepic.io/
Indices and tables

- genindex
- modindex
- search