Team 9 Final Project: Project Design

Lost & Found Anonymous Service

Due Date: 10th November 2015

Code Repository

[Final] Tech Stack Design (metal up):

[Operating System]	**AWS Ubuntu 14.04 LTS Instance**
[Load Balancer]	**Amazon ELB**
[Application Server]	**uWSGI + nginx**
[Application Framework]	**Django**
[Relational Database]	**MySQL**

[Final] Backend Roadmap (metal up):

- [**Operating System Setup**] Deploy **3 Ubuntu 14.04 LTS instances on AWS**
- [**Load Balancer Setup**] Begin the **Amazon ELB** service for those 3 instances
- [**Server Setup**] Create setup script to be used on all 3 instances
    - [**Application Server Setup**] Setup **uWSGI + nginx** configurations: **link**
    - [**Database Setup**] **PostgreSQL** installation
        - One server acts as auth DB master, others act as auth reads (read servers updated upon an auth write on the master)
    - [**Application Framework Setup**] **Django** installation

[Final] Application Roadmap (MVC):

- **[Model]** User Model
    - Will have to subclass auth.AbstractBaseUser to add extra fields fields, or a mirror User object for the application
    - [int] [unique] [Auto inc] [Primary Key] [Required] **userID**
    - [varchar (60)] **email**
    - [list of items owned] (ORM handles this, makes an owned_items table)
- **[Model]** Item Model
    - [int] [unique] [Auto inc] [Primary Key] [Required] **itemID**
    - [varchar (255)] QR_code
    - [varchar (30)] [Required] **name**
    - [User] [Required] [Foreign Key] **owner**
    - [small int] **status** (0 = Recovered, 1 = Found, 2 = Lost)
    - [boolean] **isPublic** (default == true)

- **[Controller]** User Auth (Registration, sign-in, sign-out, password reset, secure password storage)
- **[Controller]** Database lookup for each page requiring it, see below in routes if required
- **[Routes]** The URIs that the web application responds to

- /, home landing page
    - links to /login
    - link to /register
- /profile/, profile page of user, displaying all registered items
    - May **set** a previously registered item's status
        - **[DB Write]** Item status may be **Lost** (unknown location)
        - **[DB Write]** Item status may be **Recovered** (In possession of original owner)
        - **[DB Write]** Item status may be **Found** (In possession of 3rd party) (automatic)
    - **[DB Read]** Displays a **list of owned items** with links to each one
- /items/, a page where a logged in user may add new items
    - May **add** a new item to be registered, form includes:
        - **[DB Write]** Type of item
        - **[DB Write]** Name of item
        - **[DB Write]** Picture of item (to confirm ownership) (2MB max)
        - **[DB Write]** After new item is registered & saved to db
            - **[Send Email]** Sends an email to the user with a QR code to be printed
            - **[Application Call]** Displays a QR code to be printed
                - QR code is created using a python library like "qrcode"
- /items/{item_id}, sharable item link with brings up an overview of that item
    - **[DB Write]** Logged in owner may edit from here
    - **[DB Read]** Not logged in users & logged in users (non-owner) may see the item display page
    - **[DB Write]**
    - Contains a "share link" to make it easy to share the item (may need to be open-graphed optimised)
- /found/{QR_hash}/
    - **[DB Read]** show the item (brief info) and the user email details for the item owner
    - **[DB Write]** Mark item as found if it exists
- /register/, user registration page, form includes:
    - **[Auth DB Write]** First name
    - **[Auth DB Write]** Last name
    - **[Auth DB Write]** Email address
    - **[Auth DB Write]** Password
    - Password Confirmation
    - **[Auth DB Write]** Address (to ship found items to)
- /login/, user login page, form includes:
    - **[Auth DB Read]** Email address
    - **[Auth DB Read]** Password

        - **[Auth DB Read]** Redirects to /profile/ under successful authentication
    - /logout/, clears authenticated session cookie from the user's computer
        - Automatically redirects to / after cookies are cleared
- **[Views]** Frontend that is displayed towards the user
    - Twitter Bootstrap for common CSS base
    - (If time is available) bring in a frontend framework