Web Programming Report(ECS639)
Team number: 6 ------- Mohammed Owais Khan, Muhammad Nabil Fadhiya, Mohanad Al Sayegh

Additional features added: (1) Use of Mysql and (2)REST API
Mysql -Feature

Backend Implementation.
Deployment on OpenShift had the following extra features:

- MySQL Database Implementation
- phpMyAdmin Interface add on

MySQL Database Implementation:
MySQL Database implementation involved adding a cartridge via the OpenShift interface followed by configuring the settings.py file (daily_bugle.settings) to connect Django with MySQL.
One issue we faced with the implementation was the outdated python package for connection to MySQL (which was PyMySQL v0.6.2). We were required to use mysqlclient v1.3.12 to connect the MySQL database.
phpMyAdmin Interface add on:
Implementation of this add on only required us to add a cartridge via the OpenShift interface. This interface allows us to communicate directly with MySQL to read/modify tables within the databases.
Accessing the Application:
The following details are URLs and credentials for this web application.

Description	Details
URL	http://dailybugle-nabil.apps.devcloud.eecs.qmul.ac.uk/
Superuser: Username	admin@admin.com
Superuser: Password	admin12345
phpMyAdmin: Username	adminB4Yn327
phpMyAdmin: Password	rusk61d9e-MV

Rest Framework-Feature
Using Django's view set and the sterilises I was able to connect the api to the respective models. For instance, model 'User' had a viewset called 'UserViewSet' this will provide all the user records and link the UserSerializer to a serializer . In this case it was UserSerializer which will provide all the meta information e.g 'url', 'email', 'articles' Finally the router will link the url to the appropriate view set when used. Using the HyperlinkedModelSerializer made every relationship into a hyperlink.
The entry point (/api) will include hyperlinks for each model. Clicking on them will provide a list of the respective records, the following provides an overview for each hyperlink on the entry point of the api.
API:
Entry-point - /api
List of Articles: /api/Articles
List of users: /api/User
List of Comments: /api/Comment
List of Category: /api/Category
Design:
For example:
API entry point: (/api) --→ List of articles---→individual article(self link) --→ links to author, comment, category items.
The entry point includes will show each models link because other client api's may take advantage of only requiring certain information instead of routing down a specific path like Category → List of articles-→article --→author.
Rest api:
I have designed it to be more like a rest api by including self-link relation on each individual item, as well providing links for each resources instead of a read-only value, making it easier to browse for a web application and web service

Access to api:
I have restricted access to the rest api, by only allowing the super user to view access the rest api, this will avoid any authenticated user accessing any personal information of other users. The Superuser credentials are above.