Flat Organizer

Benjamin Behm
Daniel Tandberg Abrahamsen
Haakon Sønsteby
Team Nordic
itech
2077473b, 2054832a, 2056007s
{ 2077473b, 2054832a, 2056007s}@students.glasgow.ac.uk

ABSTRACT

Flat Organizer is a web application that helps people to manage tasks in households. It provides a simple interface, where users can keep track of tasks that need to be done. The main page contains two different lists: a continuous chores list, and a shopping list. A continuous chores list is meant to be for tasks that does not expire, but can be done regularly. A shopping list is a list of items that should be bought to the household. In addition, Flat Organizer provides a simple view of what each member of the household has done and when. Each task will be scored based on how challenge it is to perform. Users' points will increase as they perform tasks.

1. AIM OF APPLICATION

Most people lives in one a kind of household, it might be with their family, partner, roommates or perhaps some sort of student accommodation. While living in a household there are obligations among the residents that have to be fulfilled. Flat Organizer will make sure that each person in the household will contribute to the fellowship by both listing things that have to be done, and maybe even more important, shows a record of what everyone has completed in the last period of time.

1.1 Functional Requirements

We used so called MoSCoW technique to prioritize required functionalities. There are four different categories:

- **M** = Must have
- **S** = Should have
- **C** = Could have
- **W** = Would not have

ID	Name	Priority
FR-1.1	Show all flats a user is a member of	M
FR-1.2	Select a flat, that the user is member of	M
FR-1.3	Create a new flat, providing a flat name, description	M
FR-1.4	Invite members to flat	M
FR-1.5	Allow invited members to join a flat	M
FR-1.6	Edit the name, description and members of a flat	S
FR-1.7	Delete a flat	S
FR-1.8	Allow users to leave a flat	S
FR-2.1	Show all tasks related to a flat	M
FR-2.2	Create a new task, by specifying name, description and category	M
FR-2.3	Select a task, and set a task as done	M
FR-2.4	Delete a task	S
FR-3.1	View high score by user	S
FR-3.2	See a user's recently completed tasks	S
Fr-4.1	Create a new account, providing first name, last name, email, and password	M
Fr-4.2	Log in with their credentials	M
Fr-4.3	Log in with their Facebook/Gmail account	W
Fr-4.4	Forgotten password functionality	S

1.2 Nonfunctional Requirements

- Responsive UI, screen resolution down to 800x600 px must be supported
- The application must work on the most common browsers: Google Chrome and Mozilla Firefox
- All data must be available in the data storage even though the user deletes it.

- Security and authorization.
- All operations should be performed within a second, as after a couple seconds a user will start thinking that something is broken.
- Any personal information should not be accessed by outsider.
- Good usability.

1.3 Assumptions

The implemented application will be used on laptop or desktop as it is not designed for mobile using. This application would also be used on mobile devices, but that would be implemented afterwards if the team continue to extend the application after delivery.

1.4 Design goals

The overall design goals are allowing users to create and join flats, create and complete tasks and keep track of who did what in a flat. This should be accomplished by letting the user interact with an interface following the Keep It Short and Simple (KISS) guidelines. This is a functional, target driven application, so the number of disturbing elements should be minimized and the focus should rather be on letting the user do his/her desired task in a simplistic, neat way. Less is more.

Time is one big constraint, but managing the time and estimating the tasks so that the scope of the application is appropriate may be even harder. None of the team members has any experience of Django or Python. Therefor this is a new web development framework and programing language for all team members to become comfortable in. Some team members have a small amount of HTML, CSS and JavaScript experience, but this is very basic. When it comes to the three person team size, this is both an advantage and a disadvantage. Only three people make the project easier to manage, communication flow is easier and every group member can be involved in all parts of the system. The disadvantage is that more people would generate more inputs on ideas, most likely have more knowledge and can produce more features to an application.

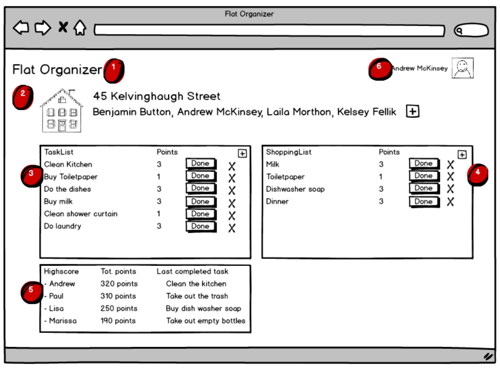
Estimating the appropriate scope of the application is challenging. Especially when the web framework is new to every team member, giving no knowledge regarding the features it supports. To try to overcome this challenge, priorities are used on every functionality implemented, so that core functionality is in place and more features can be implemented if time. The team realizes that the selected scope is ambitious, but feels that it it needs to be of this size in order to work as a complete, usable application.

2. CLIENT INTERFACE

2.1 Wireframes

2.1.1 Main page

1. Logo of the application that works as a link back to the flat overview.
2. Description of the flat, containing a picture, name of the flat, and the current members. The plus icon will show a small text field that is used to invite new members
3. Contains a list of the continuous tasks that needs to be done inside a flat. Every task has credits, a done button and a delete icon.
4. Shopping list contains the items that only needs to be done once. Every item has a name, credits, a done button and a delete icon.
5. The highscore list will show all the users, with their total points and last completed task, giving a quick and easy view of the score in the flat.
6. The users name and profile picture is clickable and gives access to a dropdown menu, giving the options to sign out.



TaskList	Points		
Clean Kitchen	3	Done	X
Buy Toiletpaper	1	Done	X
Do the dishes	3	Done	X
Buy milk	3	Done	X
Clean shower curtain	1	Done	X
Do laundry	3	Done	X

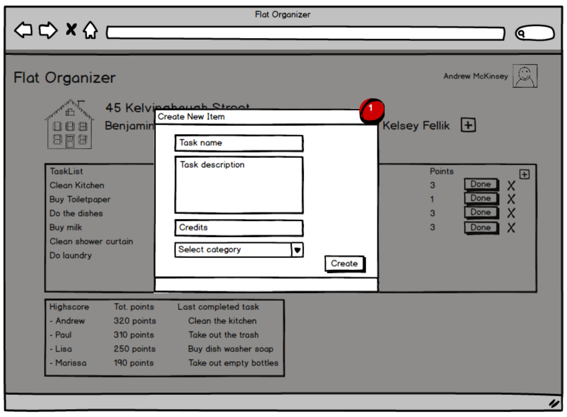
ShoppingList	Points		
Milk	3	Done	X
Toiletpaper	1	Done	X
Dishwasher soap	3	Done	X
Dinner	3	Done	X

Highscore	Tot. points	Last completed task
- Andrew	320 points	Clean the kitchen
- Paul	310 points	Take out the trash
- Lisa	250 points	Buy dish washer soap
- Marissa	190 points	Take out empty bottles

Figure 1: Main page

2.1.2 Create an item

1. By clicking on the upper right plus sign on either task list or shopping list a modal window will pop up, allowing the user to create a new item by filling in the appropriate values.



TaskList
Clean Kitchen
Buy Toiletpaper
Do the dishes
Buy milk
Clean shower curtain
Do laundry

Points		
3	Done	X
1	Done	X
3	Done	X
3	Done	X

Highscore	Tot. points	Last completed task
- Andrew	320 points	Clean the kitchen
- Paul	310 points	Take out the trash
- Lisa	250 points	Buy dish washer soap
- Marissa	190 points	Take out empty bottles

Figure 2: Create an item

2.1.3 *Flats*

1. Logo of the application that works as a link back to this view
2. A list of all the flats the user is currently a member of, giving the picture, name and other members.
3. A list of flats the user has been invited to join.
4. The new flat button will trigger a modal window to be displayed and the user can fill in the appropriate information.
5. The users name and profile picture is clickable and gives access to a dropdown menu, giving the options to sign out.

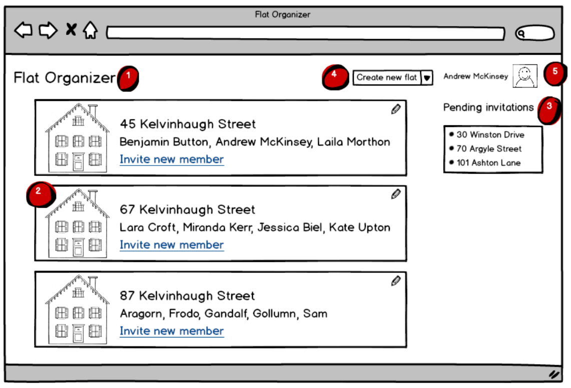


Figure 3: Flats

2.2 Use case descriptions

Use case: Create a new shopping item.
Summary: User can create a new shopping item that appears in shopping list.
Actors: User
Preconditions: User needs to be a member of the flat.
Goal of the primary user: User wants to remember what (s)he needs to buy when in shop.
Basic sequence:

1. User selects the appropriate flat.
2. User clicks on the upper right plus sign in the shopping list area.
3. User fills in a name, description, number of credits, and selects "Shopping list" from the drop down menu.
4. User submits the item.
5. The item pops up in the shopping list.

Exceptions:

- **Step 4**: User types wrong values on the form and get notification. Submit is not accepted.

Postconditions: New shopping list item appears on the shopping list.

Use case: Set shopping item as done.
Summary: After buying an item that has been on shopping list, a user want to set it done.
Actor: User
Preconditions: There is an item on shopping list.
Goal of the primary user: User wants to show that (s)he has participated in the work of a common flat and bought an item that was on the shopping list.
Basic sequences:

1. User selects the appropriate flat.
2. User finds the shopping item and clicks the "done" button
3. The item disappears from the shopping list.
4. The highscore table is updated with total score and the latest task.

Postconditions: The shopping item has disappeared from the shopping list and a user has got points equivalent with the points of done task.

Use case: Create new flat and invite Justin Timberlake.
Summary: A user has just moved in a new flat and want to create a new flat. As Justin Timberlake moved in with her, she wants to send invite to him.
Actor: User and a secondary user
Preconditions: None
Goal of the primary user: User want to create a new flat, as she has just moved in a new flat. In addition, she want to invite her friends, so that they can easily divide common tasks in the flat community.
Basic sequences:

1. User clicks the "New flat" button.
2. User fills in name and description in the modal window.
3. User clicks the "Create" button.
4. A new flat appears on the Flats page.
5. User clicks on the invite link.
6. User fills in the email address of the other member.
7. User submits the invite.

Postconditions: Invitation appears on the Flats page of the secondary user, and as he accept the invitation, his name will be shown on the Flat information of that flat and the User can also see that.

2.3 Front-end technologies

The technologies used to develop the front end is a combination of HTML5 and JavaScript, styled with the CSS toolkit Twitter Bootstrap [1]. The team used an external JavaScript library called jQuery [2] which simplified the client side scripting and provided easy access to new features. As for the discussion of implementing POST queries using Ajax, the team decided that Ajax is considered a luxury feature and will be implemented if time. All these technologies and extensions are well established with their own community and thorough documentation. There are other

CSS toolkits out here, like blueprint [3], HTML5BoilerPlate [4], and Zurb [5], but Bootstrap was chosen because of its popularity and community size.

The achievement of using all these well established technologies and libraries is a set of predefined layout guidelines and elements allowing the team to easily make use of this instead of reinventing the wheel. This helped us to make Flat Organizer a neat, clean and intuitive application, where the users immediately finds themselves at home and can conduct their preferred task quick and easily.

3. APPLICATION ARCHITECTURE

3.1 N-Tier Architecture

Flat Organizer consists of a 3-Tier layered architecture. The separation of concerns (the three independent layers) makes it easy to develop and manage. In addition, Django is based on the MVC (Model-View-Controller) pattern even though one might say that MTV (Model-Template-View) is more accurate if acronyms are important [6]. This way implementing the application and the mindset this structure forces, ensures separation of concerns.

The front end (presentation layer) is presented to the client after http requests. The clients are usually web browsers, and the requests are mostly based on URLs and form submits.

The application layer in Flat Organizer is implemented with Django, a Python Web Framework. It is the logical part of the application which handles client requests, maps URLs, performs calculations, and retrieves data from the data layer.

The data layer, the database, is a SQLite database that is a lightweight database management system and is implemented in Django applications.

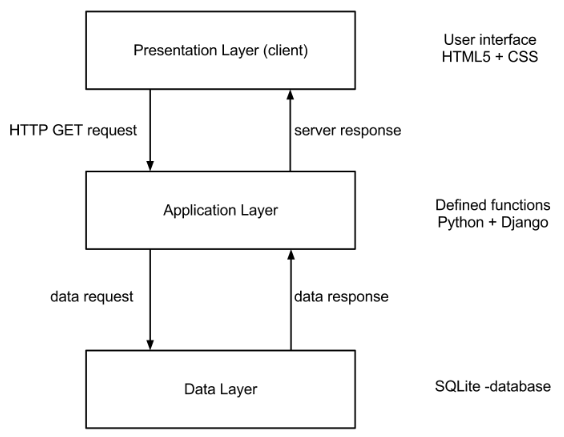


Figure 4: 3-Tier Architecture Diagram

3.2 ER Model

Figure 5 below shows the ER model, and the database is implemented as a relational database. For the implementation we used Django's SQLite support. By creating classes and relations in the model (models.py) we are able to let Django handle the requests (e.g. INSERT, DELETE, SELECT statements) with its ORM (Object Relational Mapper).

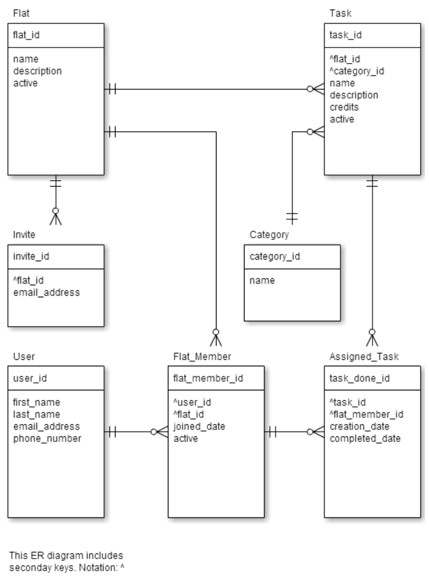


Figure 5: ER Diagram

Web Framework: Django
Django suits our project perfectly and is a recognized and easy to use Python web framework. However there are many web frameworks out there. Other frameworks are for example Flask (Python) [7], Pylon (Python) [8], Grok (Python) [9] and TurboGears (Python) [10], JSF (Java) [11], Apache Struts (Java) [12], CakePHP (PHP) [13], ASP (.NET) [14] and maybe the strongest competitor Ruby on Rails (Rails) [15], [16]. They all support N-tier architecture with focus on web development.

Web Application Frameworks focus on building web applications, and the main advantages are functionality such as session management and user authentication, data persistence and template systems [17].

Even though Web Application Frameworks have a lot of advantages there are some disadvantages that must be considered before selecting a framework. These disadvantages could be performance issues, one might need education in a specific language and framework to be able to develop efficiently, customization of functions and functionality might be hard, and if bugs and security issues are discovered, all applications that use the framework could be affected [18].

4. MESSAGE PARSING

HTTP (Web Server Request Format)
The HTTP request is sent with either of the following methods:

- POST requests for form submissions
- GET request for standards HTTP request

HTTP (Hypertext Transfer Protocol) is the standard World-Wide-Web protocol for sending messages.

Database Request Format
The connection between the database and the application is SQLite format. These transactions/requests are handled by Django.

These messages are formatted as SQL statements such as:

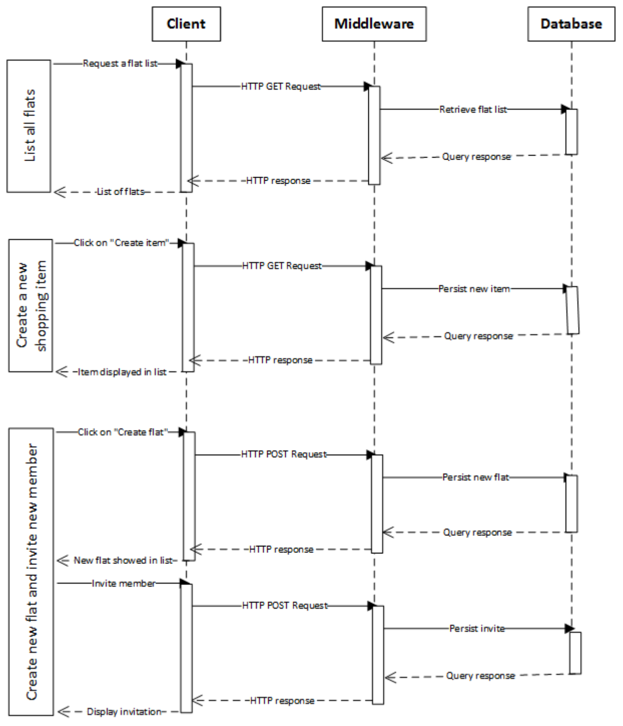


Figure 6: Sequence diagrams

```
SELECT * FROM flats_flat
```

5. IMPLEMENTATION NOTES

Views

Flat Organizer has only one view.py file, but implements many functions (redirected through the URL dispatcher). The following functions exist:

- def index(request)
- def flat(request, flatid=None)
- def profile(request, flatid=None, username=None)*
- def register(request)
- def user_logout(request)
- def user_login(request)

* Depending on a flat id, this will redirect either to the personal profile (no flat id argument), or to the users profile within one flat.

URL Mapping Schema

The following code snippet is the main URL patterns in FlatOrganizer. Every pattern redirects the client to the correct method (second argument) in views.py.

```
urlpatterns = patterns('',
        url(r'^$', views.index, name='index'),
        url(r'^register/$', views.register, name='register'),
        url(r'^login/$', views.user_login, name='login'),
        url(r'^logout/$', views.user_logout, name='logout'),
    url(r'^(?P<flatid>\d+)/(?P<username>\w+)/$',views.profile, name='flatuser'),
        url(r'^profile/$', views.profile, name='profile'),
        url(r'^(?P<flatid>\d+)/$', views.flat, name='flat'),
```

The code snippet above results in the following URLs (after: http://servername/flats/: index, register, login, logout, <flatid>, <flatid>/<username>, profile.

External Services

We did not use any external services (only serviced within the Django environment).

To be able to send email, a SMTP server was needed. During the demonstration we used Gmail's SMTP server and one personal Gmail account. However we needed only to add the account settings to settings.py and from there, Django packages handled the rest.

Functionality Checklist

- List all flats a user is a member of
- Create a new flat
- Invite members to flat by email
- Allow invited members to join a flat
- Edit the name, description and members of a flat
- Allow users to leave flat
- Delete a flat
- Show all tasks related to a flat
- Create a new task inside a flat
- Select a task, and set a task as done
- Delete a task
- View overall high score in a flat
- See a user's recently completed tasks
- Inspect roommates' profiles
- Edit personal informations
- Create a new account, providing first name, last name, email, and password
- Log in with their credentials

Known Issues

There are no known issues (bugs), but take a look at the summary and future work section for functionality that could and perhaps should be improved.

Technologies

Technology	Description	Category
Python	Object oriented and dynamic programming language.	Programming Language
Django	High-level Python Web framework based.	Web Framework
HTML	Creates web pages.	Markup Language
CSS	Describes the presentation semantics of the markup language.	Style Sheet Language
JavaScript	Interpreted programming language. Used as a dynamic script language to alter DOM.	Programming Language
jQuery	Makes HTML document traversal and manipulation easier. Multi and cross browser support.	JavaScript Library
Bootstrap	A collection of tools for creating and styling web pages.	CSS, JavaScript extension

Even though Ajax is not implemented, Flat Organizer would benefit of it. The implementation of Ajax was planned but with a lower priority (it was considered a luxury feature and was given a lower priority). Ajax (Asynchronous JavaScript and XML) consists of JavaScript and the object XMLHttpRequest, and is actually not a separate technology. Even though the messages originally were sent as XML, the JSON format is more and more used. Ajax is often used in web development at the client side to create asynchronous messages (messages that are sent in background) to the application layer. These messages are sent to the server side without the need of refreshing a page. JavaScript or jQuery is then often used for altering the DOM with new data (the response). A good tool for implementing Ajax within the Django Framework is Dajax [19].

6. REFLECTIVE SUMMARY

What have we learnt through the process of development?

We learned a lot about different web development technologies, such as HTML5, CSS, JavaScript, Python, and Django. These were quite a new area of study for most of us. In addition, none of us had never used Python or Django before. This course was extremely beneficial for us and gave us readiness to develop web application using previously mentioned techniques. We are sure that we will use Python/Django combination in future, as it provide a simple and logical way to implement web applications.

This project increased our understanding of web applications in general. Now we have better understanding of how the information flows in web applications, and what the overall structure is. Furthermore, during the development process our understanding of how different technologies and protocols interact increased a lot.

We used Git [20] for version control management. Using Git reminded us how important it is to keep track of changes. As something went broke, it was easy to return to previous commit. In addition, two of us had not used Git for a while, so it was useful to recall how to use it, as it is widely used in software industry.

How did the frameworks help progress?

As we had designed the layout beforehand, it was easy to follow those frameworks during the development process. Even though the layout changed from original drafts, those helped to understand what needed to be done as they concretized the view of the application. For this reason, it is beneficial to draw some sort of drafts of the application as these help to imagine the entirety and estimate the progress.

Problems

We did not encountered major problems during the development process. We did not have previous experience with Python and Django, therefore every now and then some one got stuck with a technical problem. Usually these were overcome fast with the support of other team members.

During the first weeks it was quite hard to follow what other team members were doing, and what should be done next. It helped a lot as we started using Github's issue tracker, where we listed all tasks and issues that needed to be done.

Major achievements

The user interference is simple and intuitive. It is easily used and does not need time to learn how to use the most important functionalities. This is confirmed by the evaluation panel at the demo. We got good feedback from industry visitors. They especially praised following aspects:

- Neat user interface
- Complete system
- Responsiveness
- Addressing real world situation

We can agree all of previous comments, and we are fairly proud of the result we got during the course.

7. SUMMARY AND FUTURE WORK

Summary of application and its current state

The application provides a simple interface to manage tasks of different flats. As a user can live in several flats (or apartments), (s)he might want to divide daily tasks efficiently with other roommates. The application is destined for that purposes. Users can maintain two different lists: a list of continuous chores and a shopping list. The former is meant for tasks that are running all the time and these stays on the list even if it is set done. The latter can be used to list items that need to be bought irregularly and are removed from the list as they are purchased.

The application is ready and functional, and it could be used in in real life. It has quite limited amount of functionalities, but it still contains all core functionalities that are needed.

Limitations

The current name of the application and the user interface can make it look as it is only suitable for student accommodations. Of course the application can be used, for example, by families that live in their own houses and want to keep track of tasks that every family member should do.

Security aspect can be seen as a limitation, as currently user's password can be phished, as it is sent as a plain text using POST method. It should be salted all the time in order to increase security.

Plans for future development

There would be a clear market for this kind of application. We should not only focus on people living in flats, as this application could also be used in regular families. There are a huge amount of households that are struggling with coordinating daily tasks for each family members. Using Flat Organizer, parents could easily inspect what their children have done and when. Parents could encourage their children to do tasks in order to get a reward. Another idea is to implement a functionality that allows users to take a picture before and after the task is done, and attach these to the task.

There are a lot of other functionalities that could be implemented in order to increase the value for users and improve the user experience. The application could be provided as a mobile application. For this reason, we could develop a mobile application, which simplifies both task logging process and viewing new tasks. It could also contain a notification functionality, so that if a user is in a shop, (s)he would be notified when the shopping list is edited.

Different flats would usually have the same basic chores. Therefore a good idea would be to implement standards tasks for the most common ones. There could be shortcut icons for most common tasks for users to click on to create new tasks.

There are some issues relating to technology that should be fixed if the application is going to be published. Currently user password is sent as a plain text as user log in. This should absolutely be fixed. In addition, we could change the way how pages are refreshed after a form is sent. We could use AJAX in form submissions, which would also make possible to implement messages that are shown to users as they do something, for example, set task done.

In addition of previously mentioned implementation possibilities, we should put effort to implement following issues as those are usually required functionalities:

User can add a profile picture in registration, but cannot change it afterward. There is a fixed flat picture, that cannot be changed. Registration does not validate the user's email. If other user has a same email, new user will get same invitations. User cannot change a password, or get a new one if old is forgotten. User cannot filter done tasks, so the list would grow very long as time elapses. List all tasks a user has done in each of his/her flats.

Another nice-to-have functionalities could be:

- Trade in points: User can trade an amount of points to get something.
- Find flats: User could be able to find flats.
- Compare flatmates: Users could be able to compare each other.
- Badges: Users get badges as they have done some tasks or got enough points.
- Facebook connection: Send invitations and notifications via Facebook.
- Money spend in addition to credits: Show how much money each has spend.

Work distribution

Every team member has been working on every part in the system. The workload has been equally distributed, as all team members did approximately 33.3% of all work. We usually worked together and used pair programming.

8. ACKNOWLEDGEMENTS

We want to thank the lecturer for inspiring lectures and giving good advices during labs. We want to thank industrial visitors, who gave us valuable feedback during a demo session how to improve the application.

References

[1] http://twitter.github.com/bootstrap/.

[2] http://jquery.com/.

[3] http://www.blueprintcss.org/.

[4] http://html5boilerplate.com/.

[5] http://foundation.zurb.com/download.php.

[6] https://docs.djangoproject.com/en/dev/faq/general/.

[7] http://flask.pocoo.org/.

[8] http://www.pylonsproject.org/.

[9] http://grok.zope.org/.

[10] http://www.turbogears.org.

[11] http://www.oracle.com/technetwork/java/javaee/javaserverfaces-139869.html.

[12] http://struts.apache.org/.

[13] http://cakephp.org/.

[14] http://www.microsoft.com/web/platform/framework.aspx.

[15] http://rubyonrails.org/.

[16] http://osdcpapers.cgpublisher.com/product/pub.84/prod.29/m.1/fid%3D174194/Askins,Green-6934-RailsVsDjango.pdf.

[17] http://docforge.com/wiki/Web_application_framework.

[18] http://docforge.com/wiki/Framework.

[19] http://www.dajaxproject.com/.

[20] http://git-scm.com/.

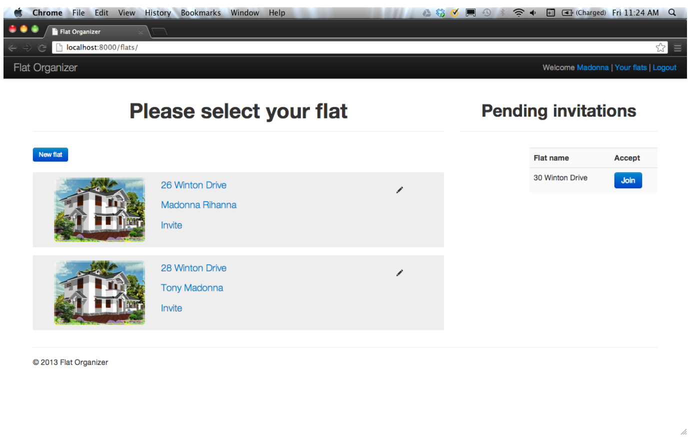
APPENDIX

A. APPENDIX: INSTALLATION

System is developed and tested using Python 2.7.2 and Django 1.4.3

1. Import the project
2. Initialize the database by running the command "python manage.py syncdb" in terminal
3. Import the populate.py script by running the command "python manage.py shell" > "import populate"
4. Start the server by running the command "python manage.py runserver". If not in a debug mode: "python manage.py runserver –insecure" to serve static files.
5. Open your browser and type in your IP address and port number (standard port number is 8000).
6. Append the /flats at the end of your ip+portnumber. Example with localhost: 127.0.0.1:8000/flats
7. You can either choose to create your own user or log in with some existing profiles:
   (a) Username1: madonna Password: madonna
   (b) Username2: rihanna Password: rihanna
   (c) Username3: gary Password: gary
8. To make the email function work you will need to add a SMTP server and provide a email host user and email host password in the settings.py on lines 174 - 178.

B. APPENDIX: USER INTERFACE



localhost:8000/flats/

Flat Organizer — Welcome Madonna | Your flats | Logout

Please select your flat

New flat

26 Winton Drive
Madonna Rihanna
Invite

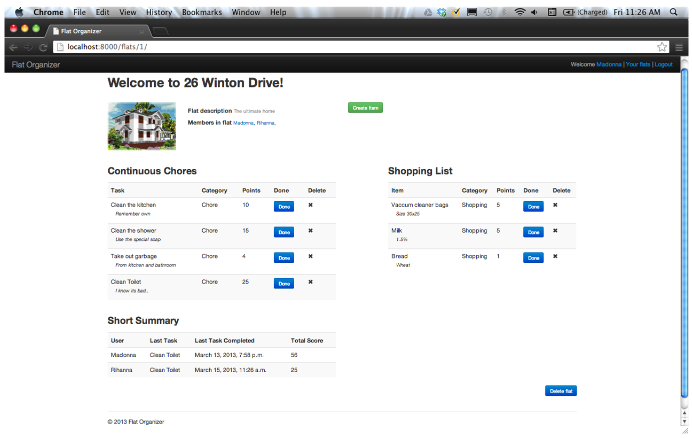
28 Winton Drive
Tony Madonna
Invite

Pending invitations

Flat name	Accept
30 Winton Drive	Join

© 2013 Flat Organizer

Figure 7: Flats



localhost:8000/flats/1/

Flat Organizer — Welcome Madonna | Your flats | Logout

Welcome to 26 Winton Drive!

Create Item

Flat description The ultimate home
Members in flat Madonna, Rihanna,

Continuous Chores

Task	Category	Points	Done	Delete
Clean the kitchen *Remember own*	Chore	10	Done	✖
Clean the shower *Use the special soap*	Chore	15	Done	✖
Take out garbage *From kitchen and bathroom*	Chore	4	Done	✖
Clean Toilet *I know its bad..*	Chore	25	Done	✖

Shopping List

Item	Category	Points	Done	Delete
Vaccum cleaner bags *Size 30x25*	Shopping	5	Done	✖
Milk *1.5%*	Shopping	5	Done	✖
Bread *Wheat*	Shopping	1	Done	✖

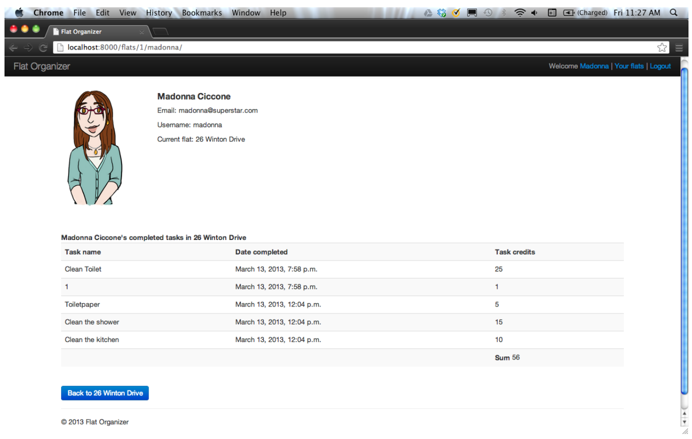
Short Summary

User	Last Task	Last Task Completed	Total Score
Madonna	Clean Toilet	March 13, 2013, 7:58 p.m.	56
Rihanna	Clean Toilet	March 15, 2013, 11:26 a.m.	25

Delete flat

© 2013 Flat Organizer

Figure 8: Main page



localhost:8000/flats/1/madonna/

Flat Organizer — Welcome Madonna | Your flats | Logout

Madonna Ciccone
Email: madonna@superstar.com
Username: madonna
Current flat: 26 Winton Drive

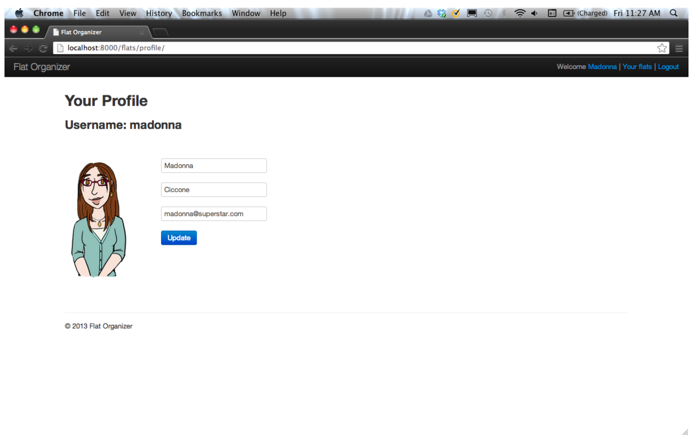
Madonna Ciccone's completed tasks in 26 Winton Drive

Task name	Date completed	Task credits
Clean Toilet	March 13, 2013, 7:58 p.m.	25
1	March 13, 2013, 7:58 p.m.	1
Toiletpaper	March 13, 2013, 12:04 p.m.	5
Clean the shower	March 13, 2013, 12:04 p.m.	15
Clean the kitchen	March 13, 2013, 12:04 p.m.	10
		Sum 56

Back to 26 Winton Drive

© 2013 Flat Organizer

Figure 9: A profile with tasks



localhost:8000/flats/profile/

Flat Organizer — Welcome Madonna | Your flats | Logout

Your Profile

Username: madonna

Madonna
Ciccone
madonna@superstar.com
Update

© 2013 Flat Organizer

Figure 10: Main profile