Memoryz database-practical work

Antti Kotiranta, June/July 2015.

1.1 Jump into the world of Memoryz

What is Memoryz?

As now days people have so many things to remember, Memoryz website is here to help working class Joe, business manager Bill, stay-a-home mother Martha, student Sandy, stock-market Stud, homeless Hank, Jeff and everyone to help them remember their important matters. The idea behind Memoryz is that users can create their own table of things needed to be done, class them, rate their priority, rank them in different order and of course mark them as done. Memoryz will provide full chore controlling system to you anywhere and anytime ;).

Memoryz is a web-site where users create their account. After logging in, users can create new chores, set their priority, create labels and add them to chores. Every user has their own private chores and labels. Chores can be marked as done and they can be edited and deleted if necessary. Also labels can be edited and deleted.

Implementation

The practical work is a web-site based interface which uses sql database. This practical work does not use any ORM libraries. The database used in this practical work is PostgreSql and the functionality between the interface and database is written with PHP.

The works environment will be implemented with Helsinki University computer science's user server, but if other environment is used the platform/environment will need to support at least PostgreSql (8.4.22) and PHP (5.3.2). At least JavaScript support is needed from the browser, but any modern browser will be sufficient.

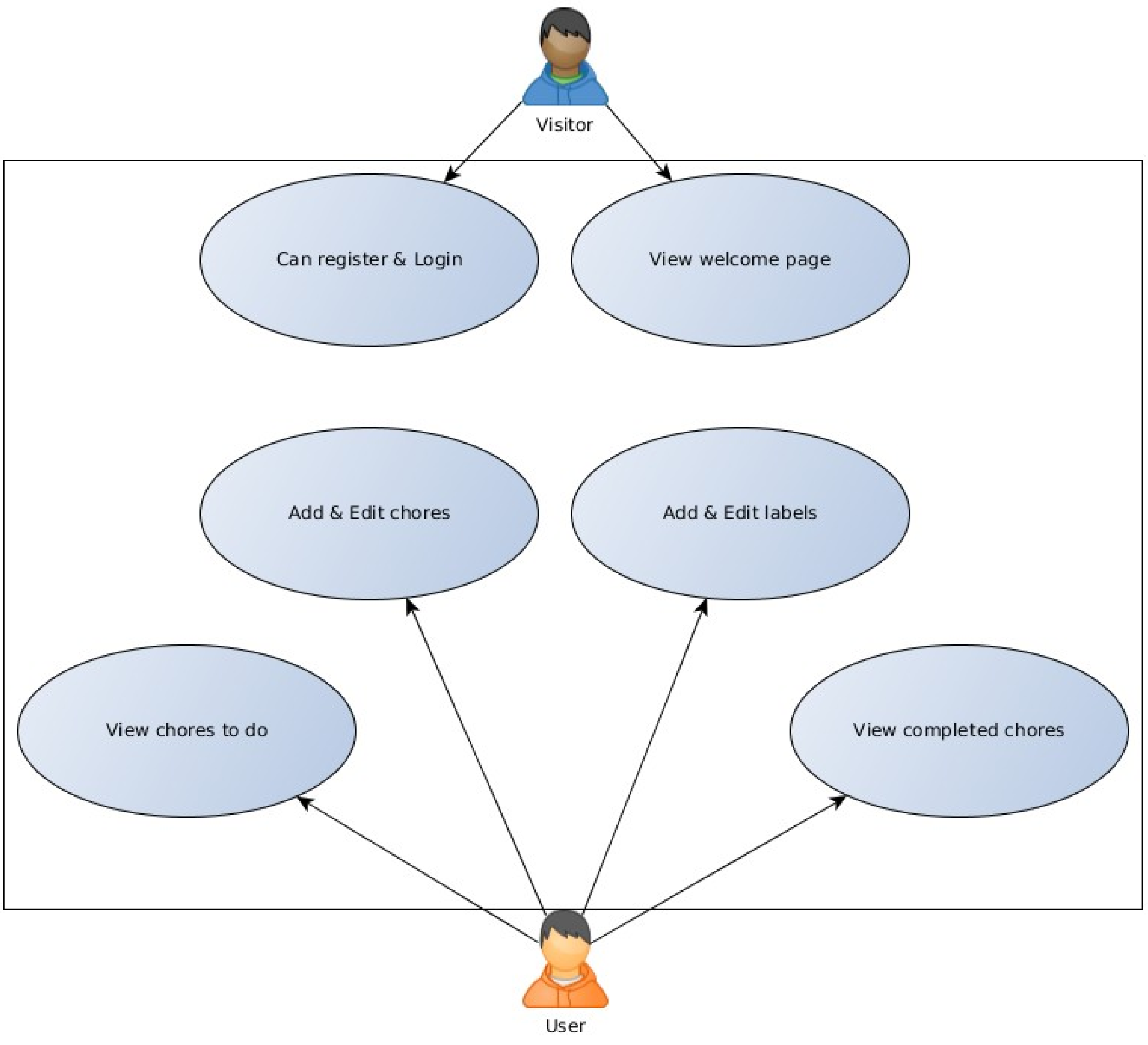
More specific info will be added later.

1.2 How to use

Log in with username= Testi2, password = Testi123.

2. Use case

2.1 Use case diagram



Visitor

Visitors is anyone who visits the site. Everyone is a visitor before logging in.

User

User is a registered and logged in visitor.

2. 2 Use case descriptions

A visitor may:

-See a welcome page which contains basic information of the site

-Register as user

-Log in

User:

Preconditions: The user is registered and logged in.

-View chores to do: The user can browse their own chores they haven't completed in a list, which includes the chores priority and labels.

-View completed chores: The user can see chores they have marked as done, it's priority and labels listed as well.

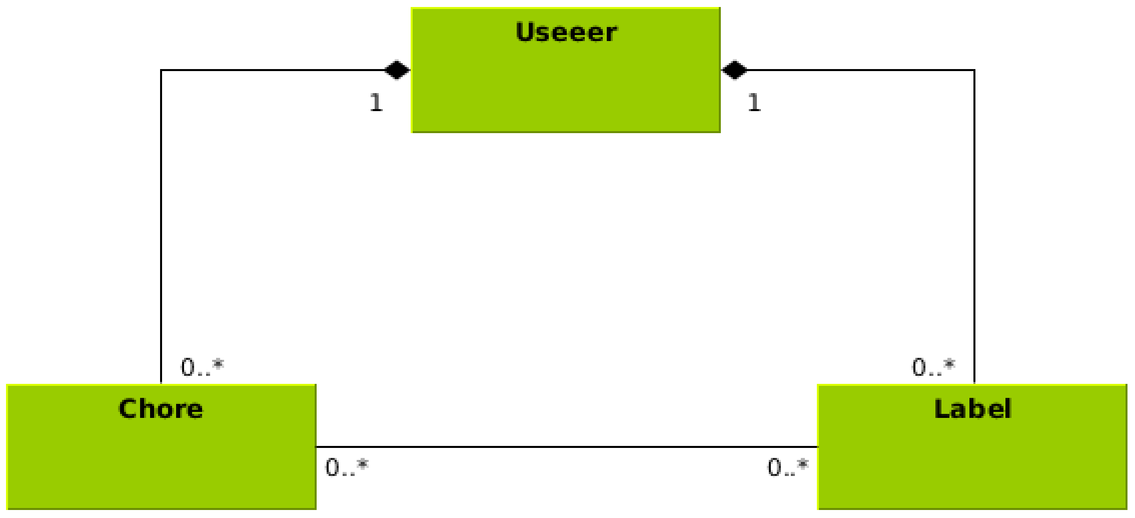
-Add & Edit chore: The user can add a new chore by giving it a name, description, priority and labels. He/She can also edit a chore and delete it if necessary.

-**Add & Edit label:** The user can add a new label by giving it a name and description. He/She can also edit a label and delete it if necessary.

-Log out.

3. System data content

3.1 Package diagram



Useeer table has only name and password columns. Useeer may have many chores and labels.

Chore

Attribute	**Value**	**Description**
Name	Varchar, max 20 characters	I.e “School practical work”
Description	Varchar, max 70 characters	Chores description, i.e. “Database practical work”
Priority	Integer, between 1-5	Marks chores priority, 1 being most important and 5 the least.
Done	Boolean	Marked if the chore is done, default false.
Deadline	Date	Chores deadline

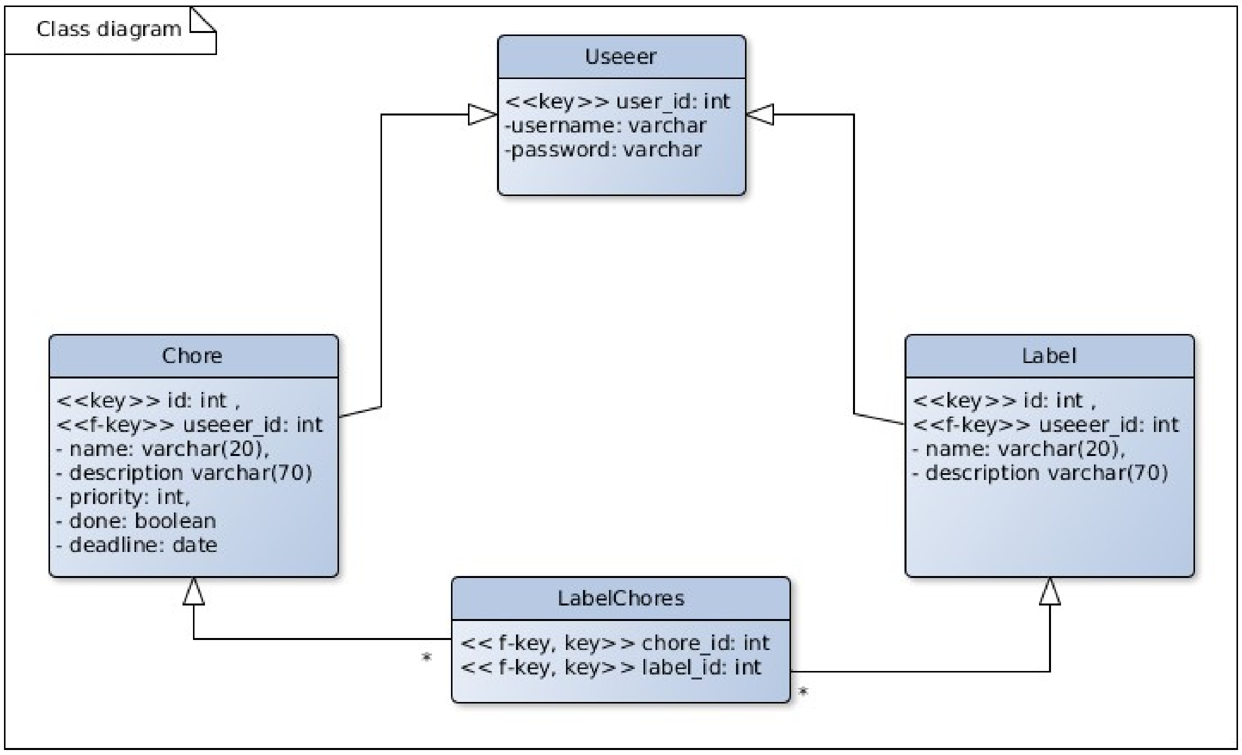
Chores belongs to a single user (Useeer). Each chore has their own content. Chore may or may not have many labels.

Label

Attribute	**Value**	**Description**
Name	Varchar, max 20 characters	i.e. “School stuff”
Description	Varchar, max 70 characters	i.e. “School related chores”

Labels belong to a single user with individual content. Label may have many chores.

3.2 Class diagram



Note: LabelChores is a Many-to-Many table between Chore and Label.

4. System content

The program is build with MVC architecture

Views can be found from app/views folder.

Controllers are found from app/controllers folder.

Models are found from app/models folder.

Routes are found from config folder.

Assets folder contains JavaScript and CSS files

Sql folder contains sql statements.

Lib contains BaseController, BaseModel and other help/background functions.

Vendor contains BootStrap and ConnectionTest libaries.

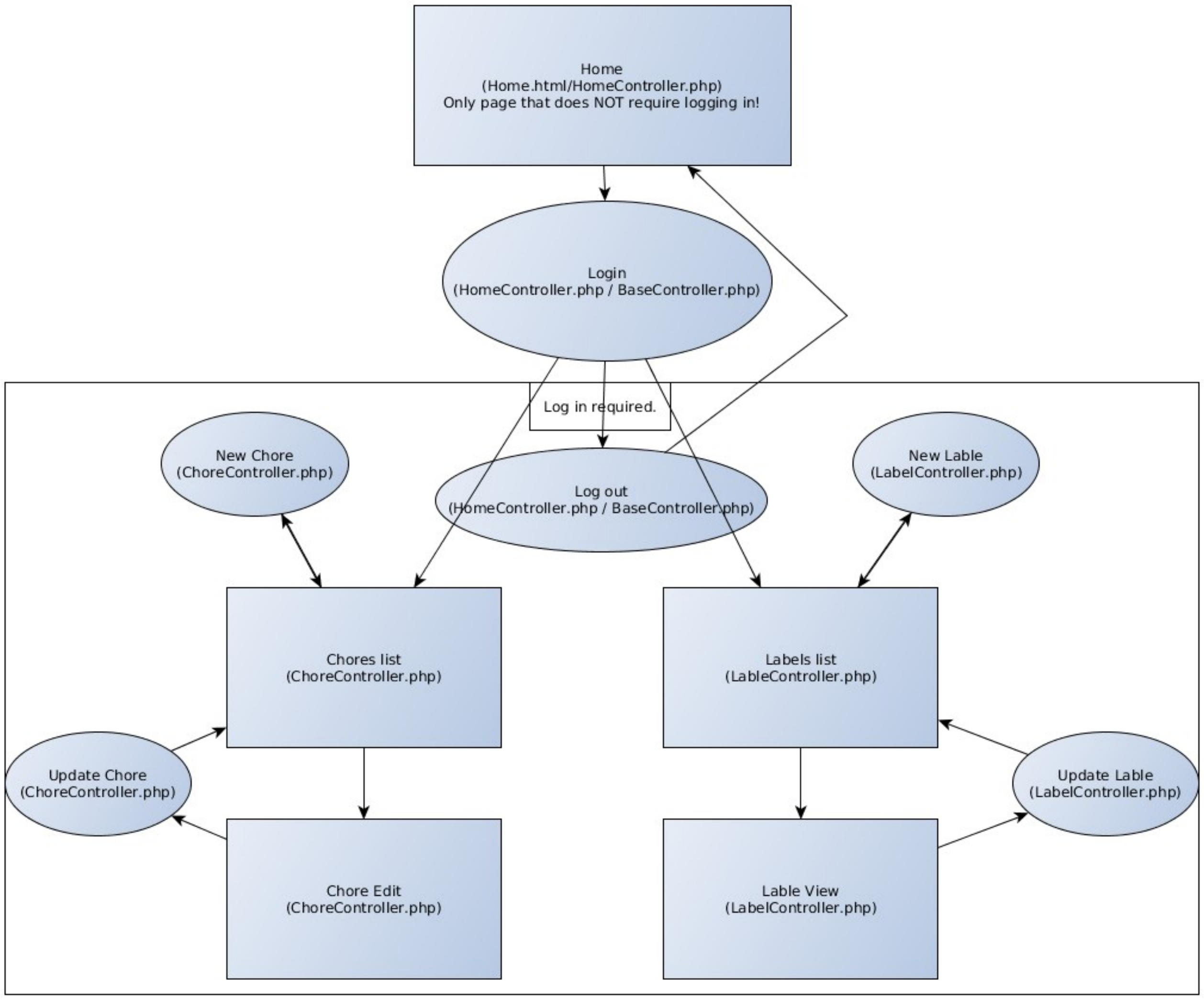
Documentation is found from doc folder.

Deploy.sh is used to deploy files to the Helsinki University users server.

Composer is used for the dependency management and composer.json is found from the root folder.

All filenames are written with lowercase letters.

5. User interface and system structure



Once user has logged in, a navigation panel to Chores and Labels lists is available.