

**Odex:**

**Experimental framework for studying the detection and generation of outlying data on high dimensional data sets**

*Design phase*

*Author:*

Ao Zuo

Christoph Usbeck

Daonan Zhang

Haoliang Huang

Yuqi Cao

10.01.2024

Catalog

[1. Introduction 3](#_Toc28021)

[1.1 Django and the fundamental structure of odex 3](#_Toc27666)

[Django 3](#_Toc2636)

[Reasons to use django 4](#_Toc22619)

[The model-view-template design pattern 4](#_Toc20263)

[Components of django 4](#_Toc30280)

[1.2 Usage 5](#_Toc4871)

[1.3 Processing logic 6](#_Toc10330)

[2. Features 7](#_Toc32734)

[2.1 Structure 7](#_Toc25673)

[2.2 Features: user management 8](#_Toc8659)

[2.2.1 TAN 8](#_Toc26054)

[2.2.2 Restore password 8](#_Toc11628)

[2.3 Features:Experiments 8](#_Toc16494)

[2.3.1 Configuration 8](#_Toc7225)

[2.3.2 Static load of parameters by Outlier Detection Method 8](#_Toc4693)

[2.3.3 Various run modes 8](#_Toc31676)

[2.3.4 Refreshing 9](#_Toc25542)

[2.3.5 Reconfiguration 9](#_Toc26779)

[3.Classes description 9](#_Toc26160)

[3.1 Module: django 10](#_Toc16485)

[3.2 Module: odex.user 10](#_Toc4955)

[3.2.1 odex.user.views 10](#_Toc9834)

[3.2.2 odex.user.forms 11](#_Toc30549)

[3.2.3 odex.user.models 12](#_Toc7294)

[3.2.4 odex.user.middleware 13](#_Toc695)

[3.3. module: odex.experiment 13](#_Toc13978)

[3.3.1 odex.experiment.views 13](#_Toc28729)

[3.3.2 odex.experiment.forms 14](#_Toc26805)

[3.3.3 odex.experiment.models 14](#_Toc24487)

[4.Sequences description 15](#_Toc10323)

[4.1 Log in and register 15](#_Toc6578)

[4.2 Restore password 16](#_Toc10139)

[4.3 Show about us 17](#_Toc19263)

[4.4 Create a new run 18](#_Toc24042)

[4.5 Show details 19](#_Toc15381)

[4.6 Delete experiment 21](#_Toc29250)

[4.7 Delete account 21](#_Toc25630)

[4.8 Log out 22](#_Toc32271)

# 1. Introduction

In this document the results of the design phase of the PSE project “Experimental framework for studying the detection of outlying data on high dimensional data sets” are compiled. The name of the software product is odex.

## 1.1 Django and the fundamental structure of odex

### Django

Django is a python web framework. It is free and open-source.

### Reasons to use django

Django is a framework that has proven itself successful in various projects. We chose django for several reasons; major reasons to use django were:

* The product owner wishes the project to be realized in Python
* Likewise, the project owners requested a web application
* Django’s follows the Model-View-Control design pattern, although it uses a variation called model-view-template which follows its own conventions
* Django includes an object relation management (ORM) which defines the database structure of an application as classes in the code
* The widespread use and general nature of django makes support and extendibility easy

### The model-view-template design pattern

Django generally uses the model-view-template (MVT) design pattern which is an adaptation of the model-view-controller pattern. In MVT there is no separate controller, however most of the functions of the controller are fulfilled by the view component.

### Components of django

*Model*: In django a model is a container for essential functions and fields. Each model corresponds to one table and each field in a model to a database field, which is a column in the database. Furthermore, django model is a subclass of django.db.models.Model. It includes an API that makes creating, retrieving, updating, and deleting from the mapped table possible.

*View*: A django view contains the business logic of the application. It is a python function that responds to user interaction with contents of a web page.

*Template*: A template consists of the static parts of an HTML page output. Dynamic content is inserted in it.

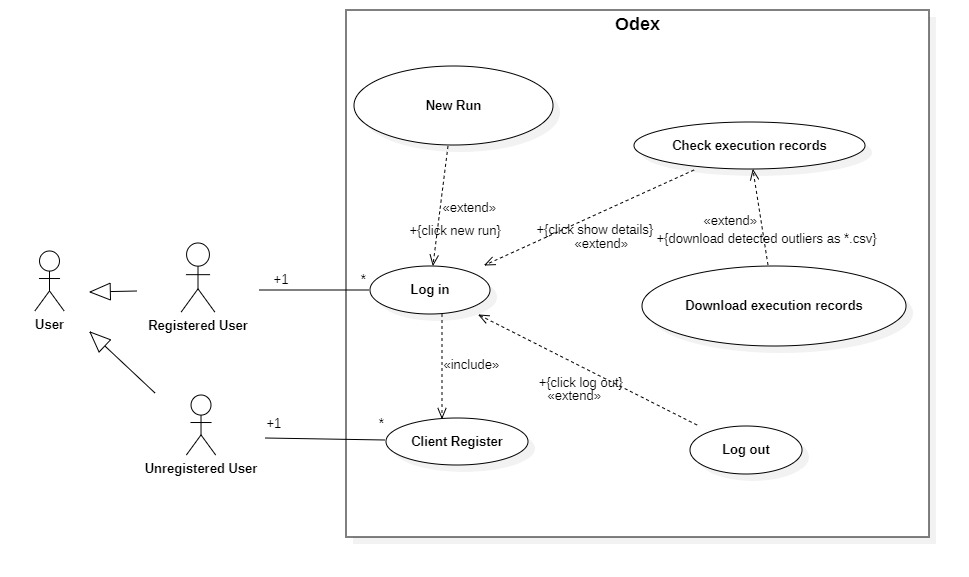
*URL mapping*: As a web framework django receives user requests in the form of a URL. URLs are handled by the **django.urls** module

*Static file handling*: Django provides functions for static files such as .css-files in **django.contrib.staticfiles**

*Model form*: Model forms are python classes which give the user the ability to make a form using python instead of having to create them in HTML, thus saving development time.

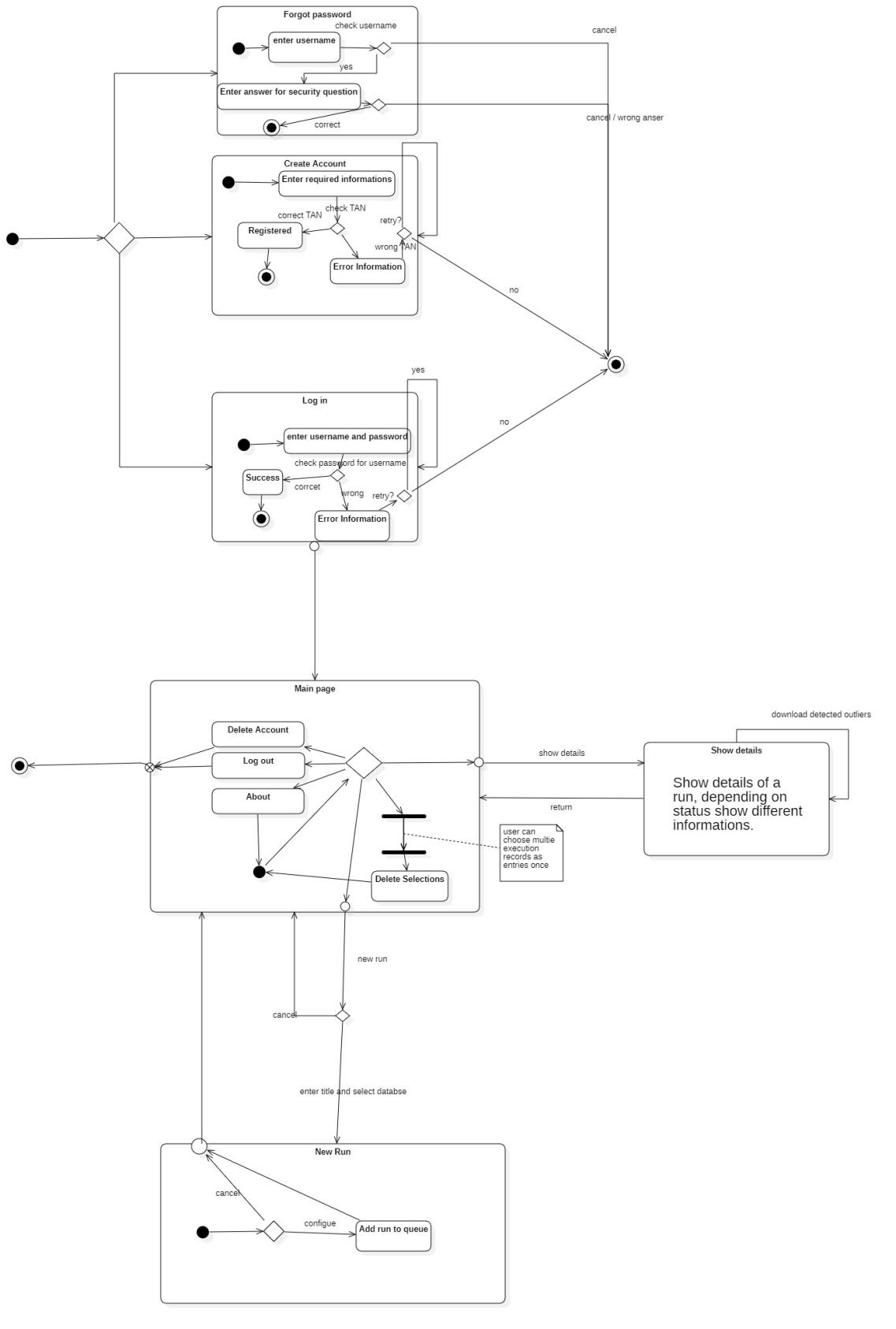
## 1.2 Usage

The application provides easy and intuitive usage for users. The following use case diagram shows the most important functions of this application: after login, users can call a new run to detect outlier, manager the records of executions or download the results.



## 1.3 Processing logic

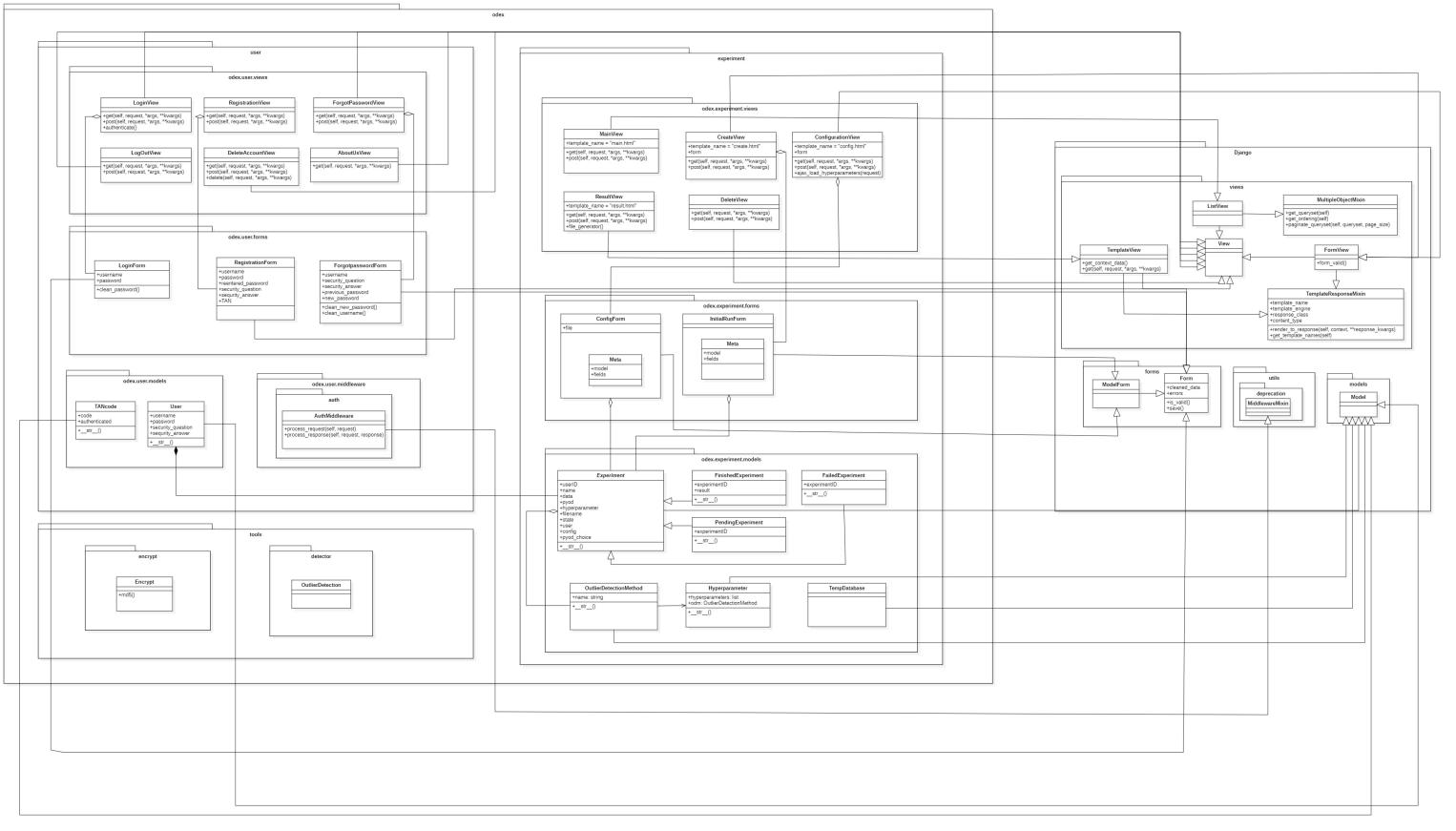
The following state diagram shows the processing logic of this application.



# 2. Features

## 2.1 Structure

Based on django framework, this application can be roughly divided into two parts: the user part and the experiment part. All account–relevant views and functions, including login, registration and password-service are in the first part. The second part includes views and functions for users after login, mainly experiment-relevant.



## 

## 2.2 Features: user management

### 2.2.1 TAN

By registering for an account, a user must give a valid TAN. Once a TAN was used, it will be set as invalid.

### 2.2.2 Restore password

The application provides a chance for the users, who forget their password, to restore it with a security question, this question should be set by registering.

## 2.3 Features:Experiments

### 2.3.1 Configuration

Based on the database from the user, the domain of subspaces can be different in the configuration page, so that the user can only choose valid legal subspaces.

### 2.3.2 Static load of parameters by Outlier Detection Method

The user will immediately see the parameters of the ODM he choosed in the configuration page.

### 2.3.3 Various run modes

The application provides users to run with various modes, users can choose as his wish if he want to run it with addtional data or compare to ground truth file or both.

### 2.3.4 Refreshing

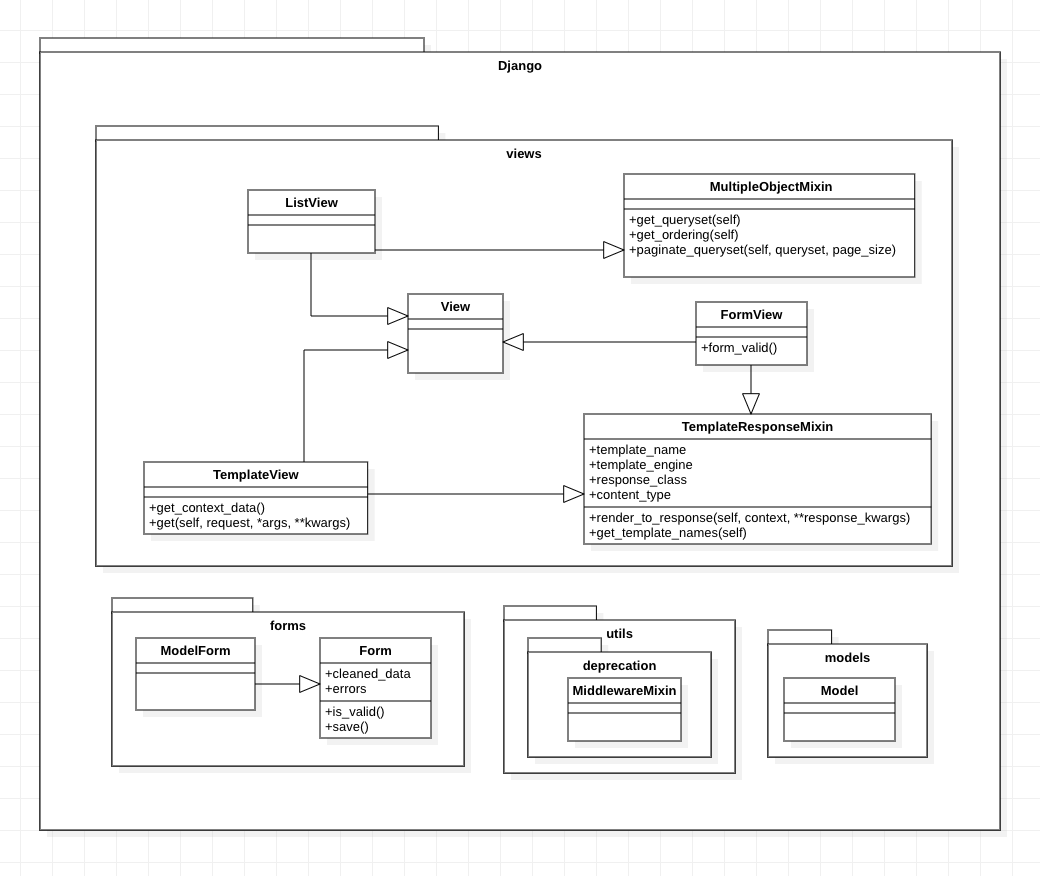
The application will refresh the main page periodically, to make sure once an execution is finished/failed, it can be seen by the user immediately.

### 2.3.5 Reconfiguration

For failed experiments, the application provides an extra button for the user, the user can click the button and get into previous configuration of this experiment to modify configs and re-run it.

# 3.Classes description

## 3.1 Module: django

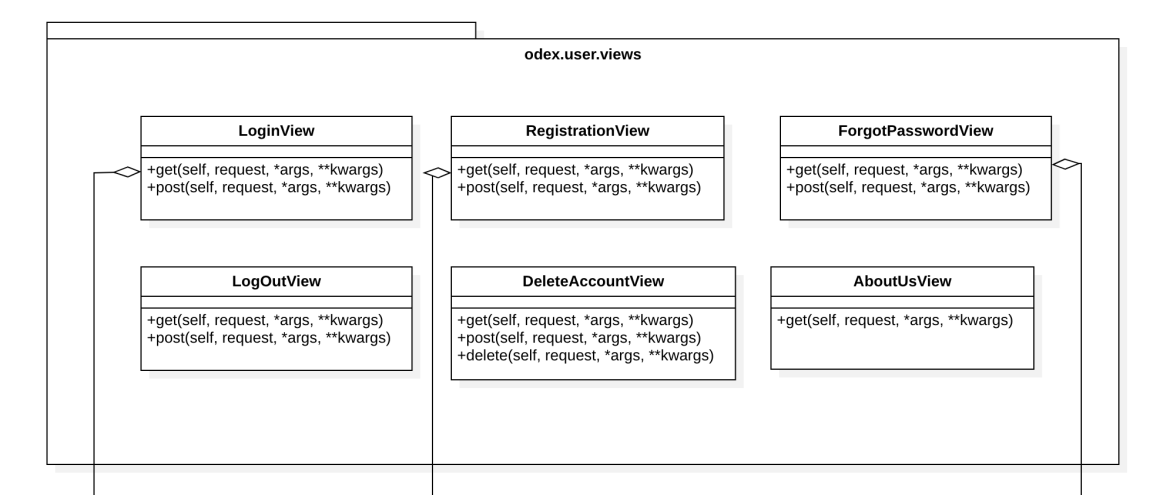


## 

## 3.2 Module: odex.user

The user module contains everything necessary for the user.

### 3.2.1 odex.user.views



**Purpose & Design**

The odex.user.views module contains django.views.View implementations (views) that have the function to provide user information.

**LoginView**

Allows a user to log in to the system

**RegistrationView**

Allows a user to register an account

**ForgotPasswordView**

Allows the user to reset password

**LogOutView**

Used to log the user out of the system

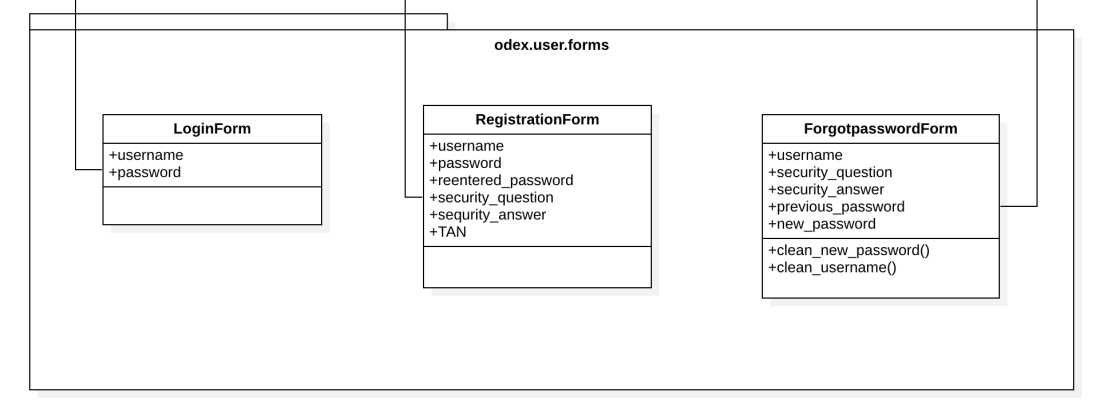
**DeleteAccountView**

Used to delete the user account

**AboutUsView**

Displays the developer information of odex

### 3.2.2 odex.user.forms



**Purpose & Design**

The odex.user.forms module contains django.forms.Form implementations (forms) through which user data may be managed.

**LoginForm**

Used to enter login information

clean\_password(): validate that the contents of a CharField called password was correct.

**RegistrationFrom**

Used to enter registration information

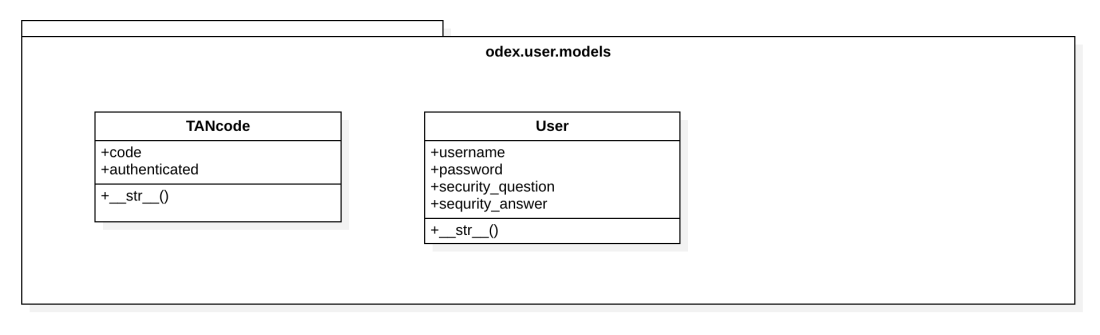
**ForgotpasswordFrom**

Used to enter reset password information

clean\_new\_password(): validate that the contents of CharField called new\_password and previous\_password were different.

clean\_username(): validate that the contents of a CharField called username was unique.

### 3.2.3 odex.user.models



**Purpose & Design**

The odex.user.models contains user and TANcode database models.

**TANcode**

The database model of a TAN code in the database

Methods:

\_str\_ (): Return the string representation of this TAN code

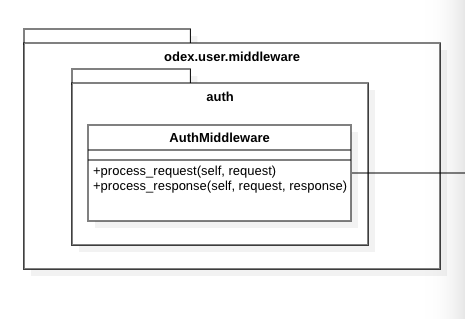
**User**

The database model of a user in the database

Methods:

\_str\_ (): Return the string representation of this user

### 3.2.4 odex.user.middleware



**Purpose & Design**

Add middleware to verify user login status to restrict access to web pages

**AuthMiddleware**

Authentication middleware that verifies whether the request sent by the browser carries a session.

process\_request(self, request): If it carries the correct session, it will continue to pass the request backwards. Otherwise, it returns the URL of login page.

## 

## 3.3. module: odex.experiment

The experiment module contains everything necessary for the experiment.

### 3.3.1 odex.experiment.views

**Purpose & Design**

The odex.experiment.views contains django.views.View implementations (views) that have the function to provide experiment.

**MainView**

Displays a list of all experiments

The user can create an experiment, delete experiments and view experiment details from this view.

**CreateView**

Used to create a new experiment

**ConfigurationView**

Used to configure the new experiment

ajax\_load\_hyperparameters(request): Used to display ajax partial refresh. If the user selects a certain ourlier detection method, it will partially refresh and display the corresponding hyperparameters that need to be entered.

**ResultView**

Displays the result of an experiment

**DeleteView**

Used to delete experiments

### 3.3.2 odex.experiment.forms

**Purpose & Design**

The odex.experiment.forms module contains django.forms.Form implementations (forms) through that experiment data may be managed.

**ConfigForm**

Used to enter experiment configuration information

**InitialRunForm**

Used to enter experiment creation information

### 3.3.3 odex.experiment.models

**Purpose & Design**

The odex.experiment.models contains all model classes that store experiment related data. This module manages database relations.

**Experiment**

Base class for experiments, inherited by experimental classes of different state.

Methods:

\_str\_ (): Return the string representation of the experiment.

**PendingExperiment**

The database model of a pending experiment.

**FinishedExperiment**

The database model of a finished experiment.

**FailedExperiment**

The database model of a failed experiment.

**TempDatabase**

Temporary storage of csv files uploaded by users

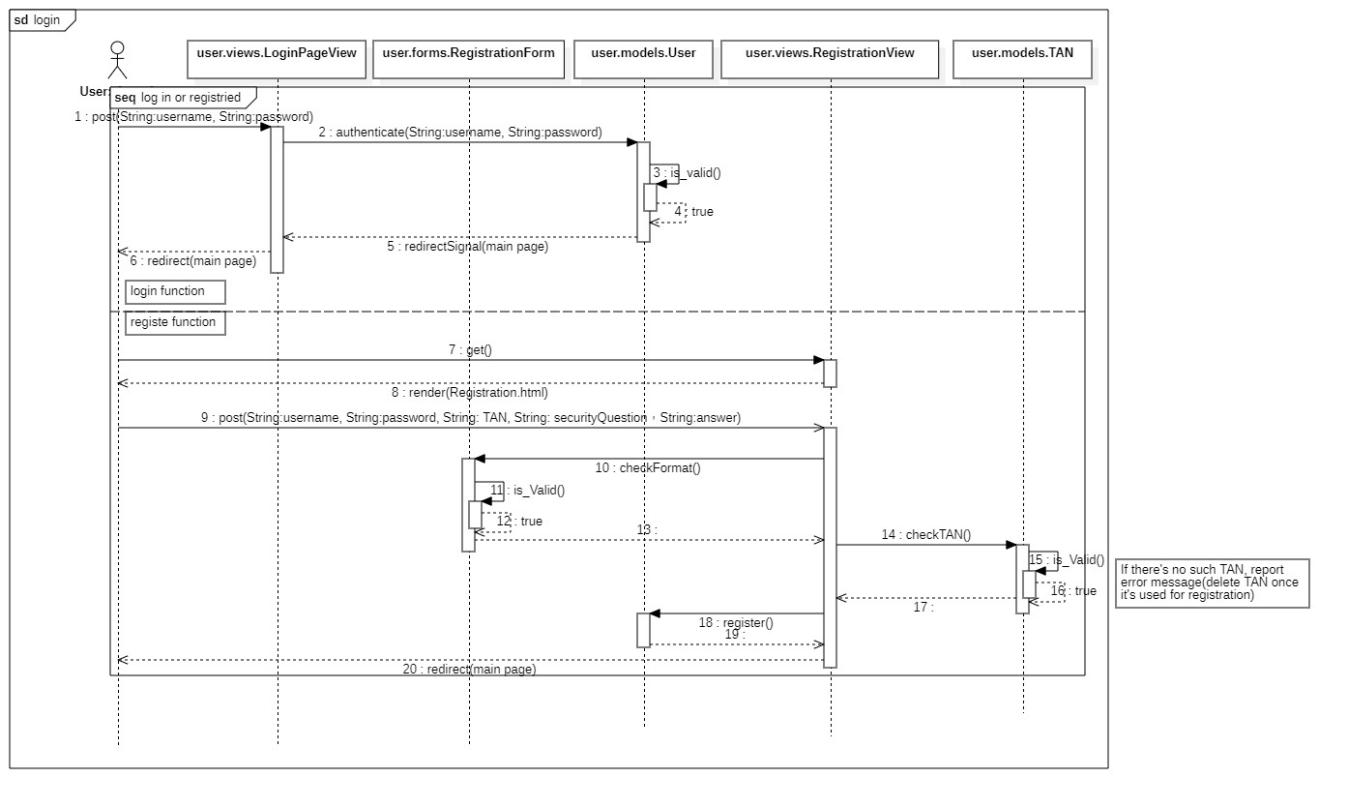
# 

# 4.Sequences description

The following sequence diagrams are intended to illustrate the essential functions of the software. It should ba apparent how it reacts to a specific request and what the output is. For this understanding it is necessary to show some Django-internal functions.

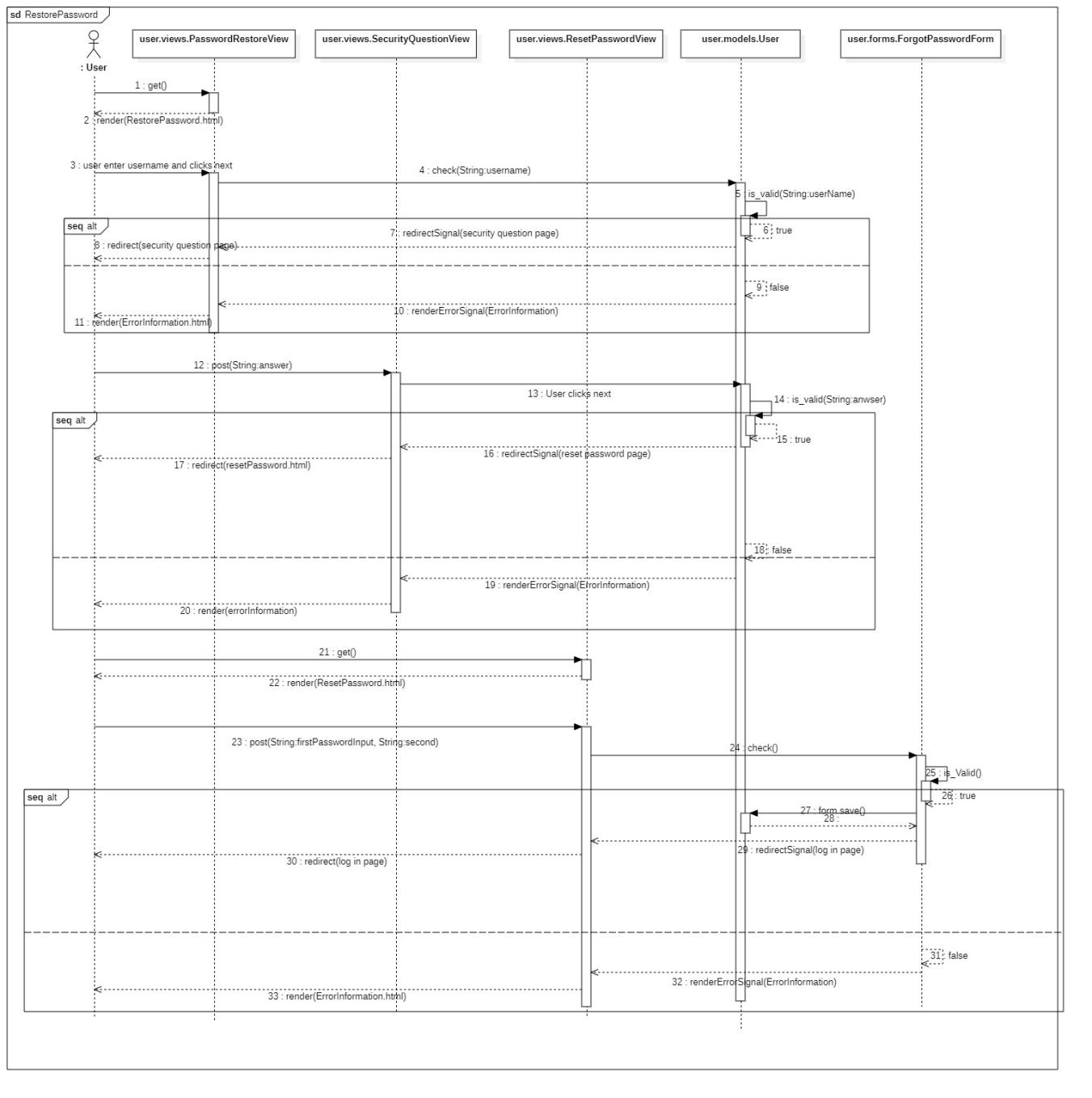
## 4.1 Log in and register

The user is located on the login page view. The user enters their username and password, and clicks “login” button. If the account is authenticated successfully, the main page view loads. If the user clicks “create account” button, the registration page view loads. The user enters a username, a TAN, a password, and a security question (chooses a security question and enters the answer to the question). If all data are valid, a new account is created and the main page view loads.

****

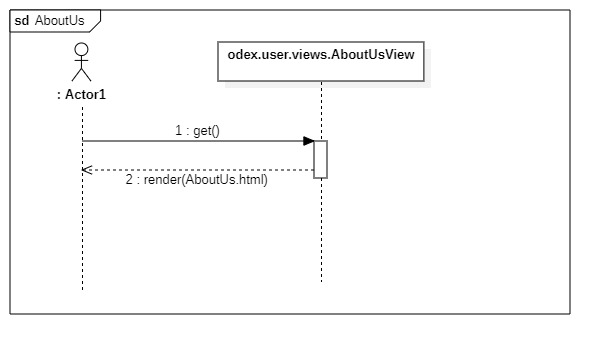
## 4.2 Restore password

On the login page, if the user clicks “forgot password” button, the password restore page view loads. The user enters their username and clicks “next” button. If the username does not exist, the reset password page view displays an error message. Otherwise, the security question Page view loads. The user enters the answer to their security question and clicks “next” button. If the answer is false, the security question Page view displays an error message. Otherwise, the reset password page view loads. The user enters the same new password twice. If the password is valid, the data is updated and the login page view loads. Otherwise, the security question Page view displays an error message.



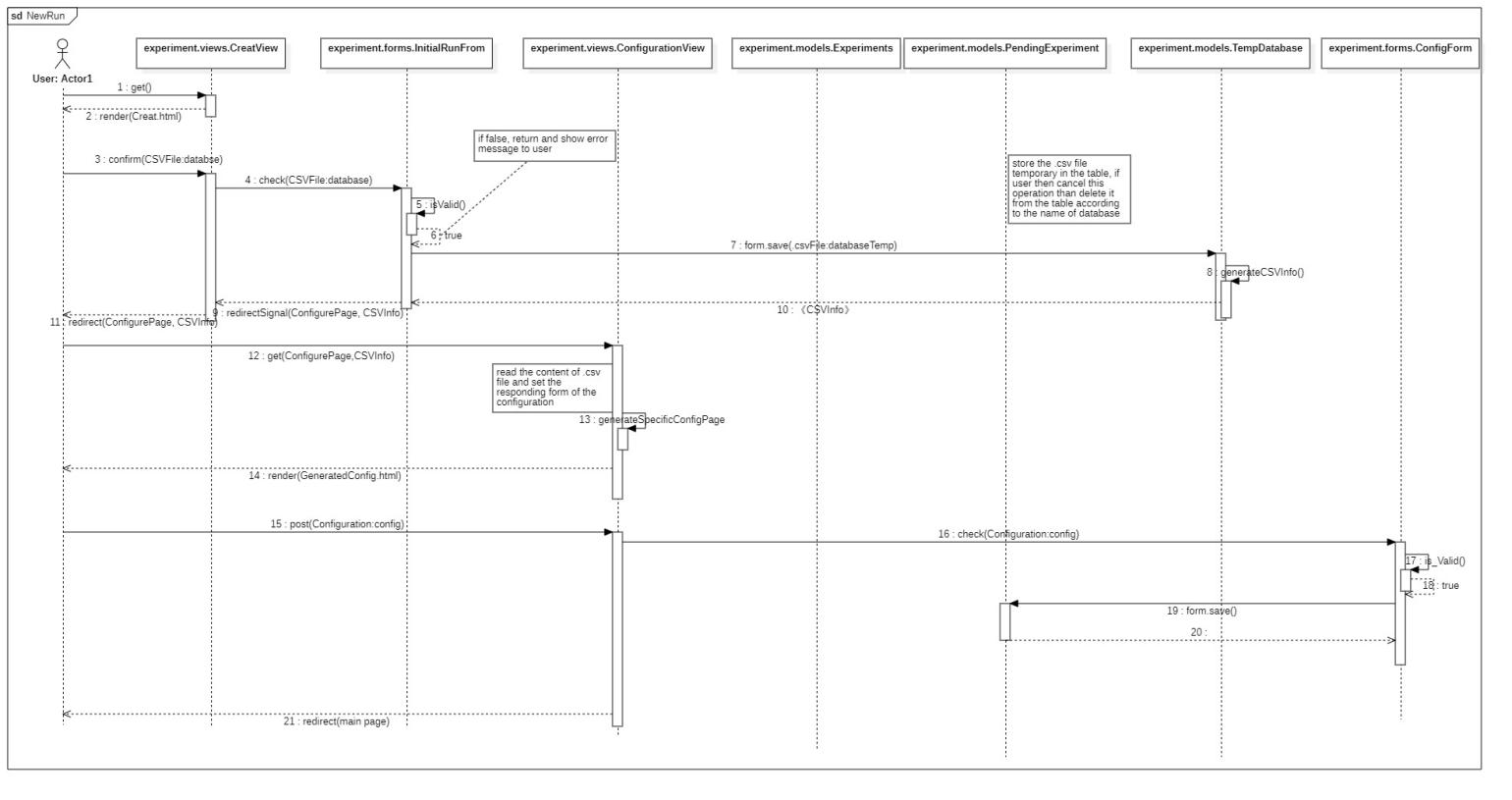
## 4.3 Show about us

On the main page there is “about us” button. The user clicks “about us” button and then the about us page view loads.

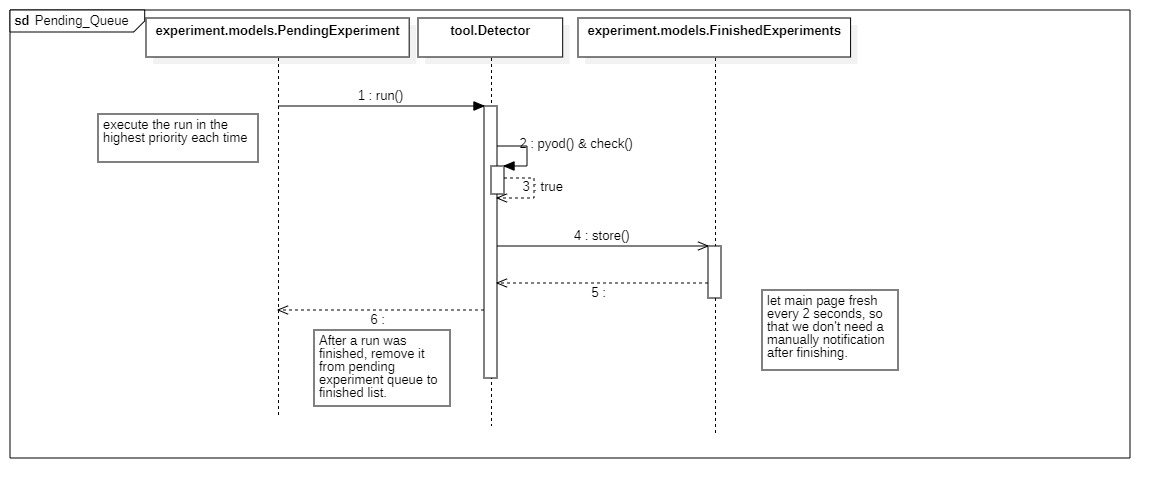


## 4.4 Create a new run

After login the user is located on the main page and clicks “new run” button. Then the create page view loads. The user uploads a database as a .csv file. If the file is valid, the file is temporarily stored in the table. Then the configuration page view loads. The column names of the .csv file are loaded on the configuration page. The user enters the configurations. If all configurations are valid, the new experiment is created and added to the pending queue. The main page view loads.

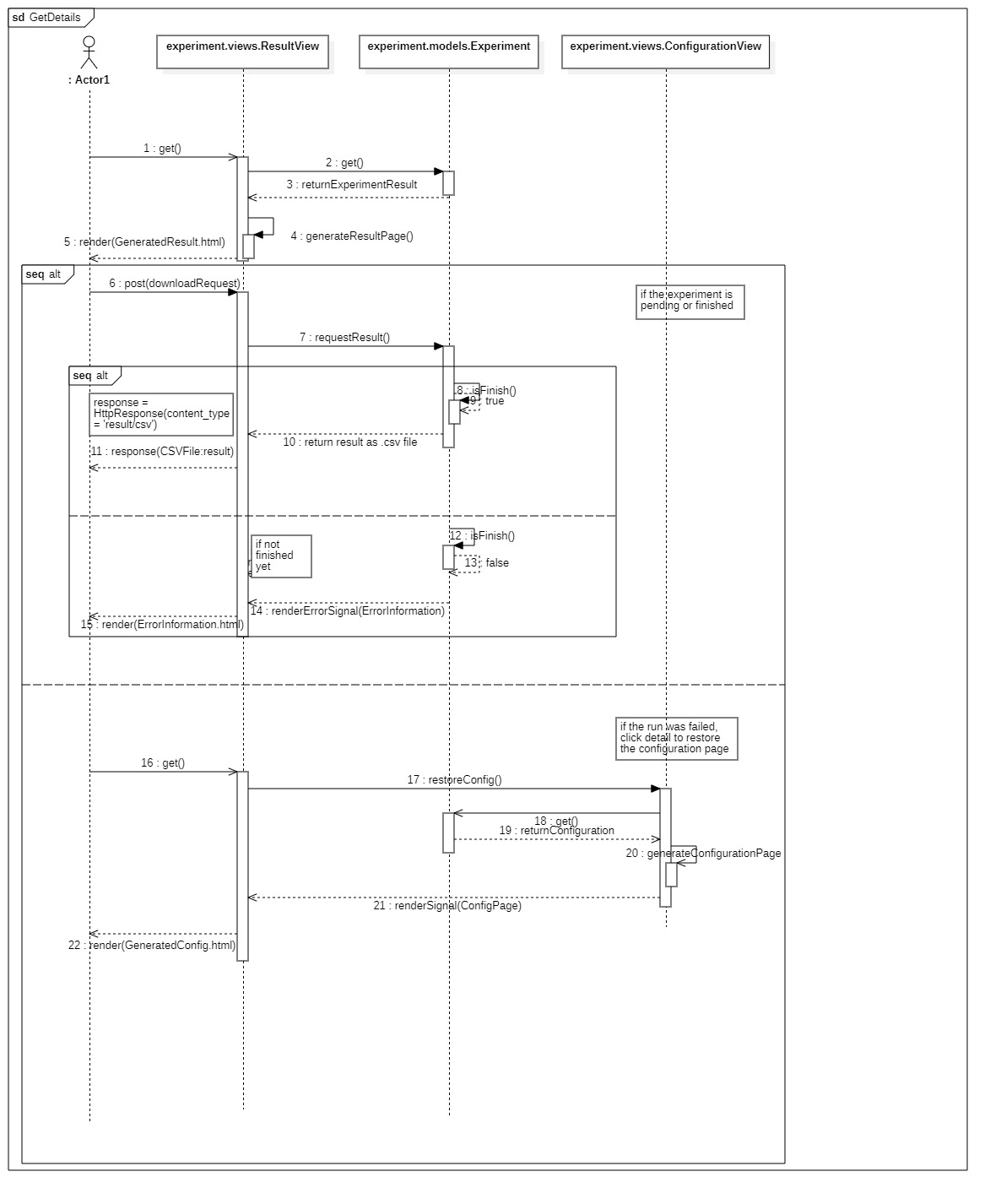


The detector executes the experiments in the pending queue. After a run is finished, remove it from the pending queue to the finished list and the result is stored in the database.



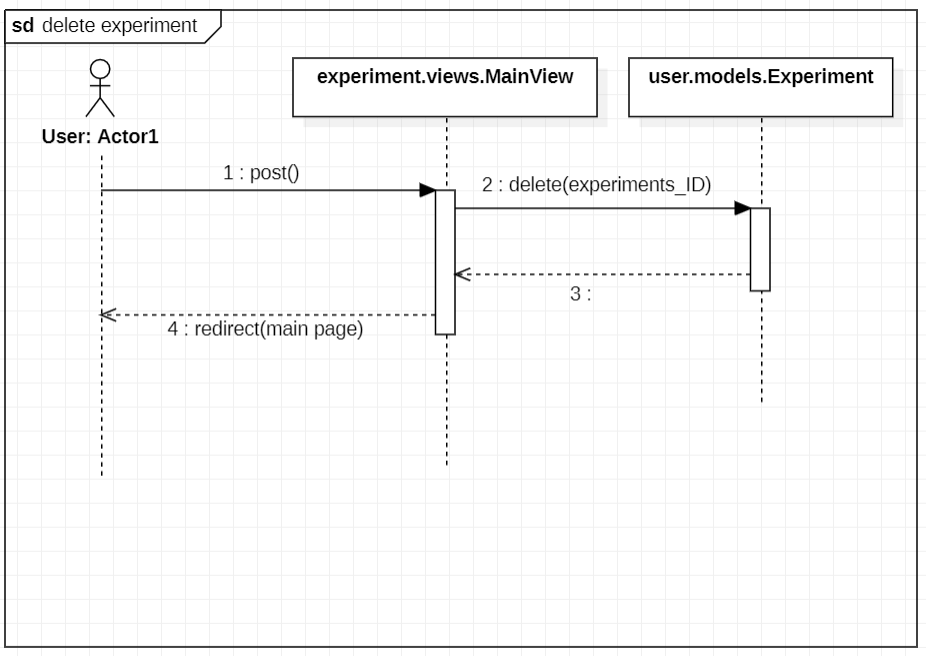
## 4.5 Show details

On the main page, there is “show details” button for each experiment. The user clicks “show details” button for any experiment. Then the result page view loads. The user clicks “download” button on the result page. If the run has finished, the user gets the result as a .csv file. Otherwise, the result page view displays an error message. If the run failed, the user can click “restore the configuration” button on the result page. Then the configuration page view loads.



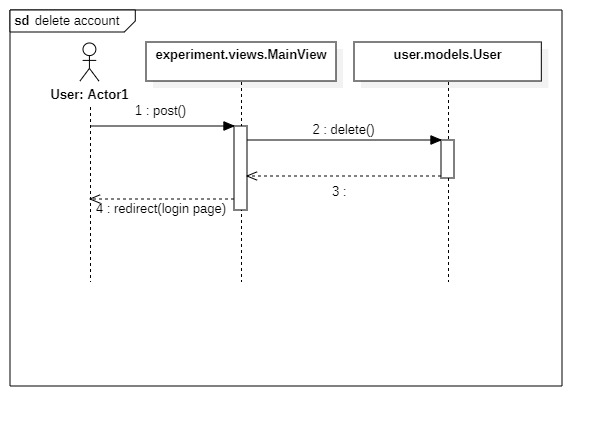
## 4.6 Delete experiment

The user calls up the main page and then selects the experiments he wants to delete and clicks the "delete" button. The experiments are then deleted from the database.



## 4.7 Delete account

The user calls up the main page and clicks the “delete account” button. The account is then deleted from the database. The login page view loads.



## 4.8 Log out

The user calls up the main page and clicks the “log out” button. The login page view loads.

