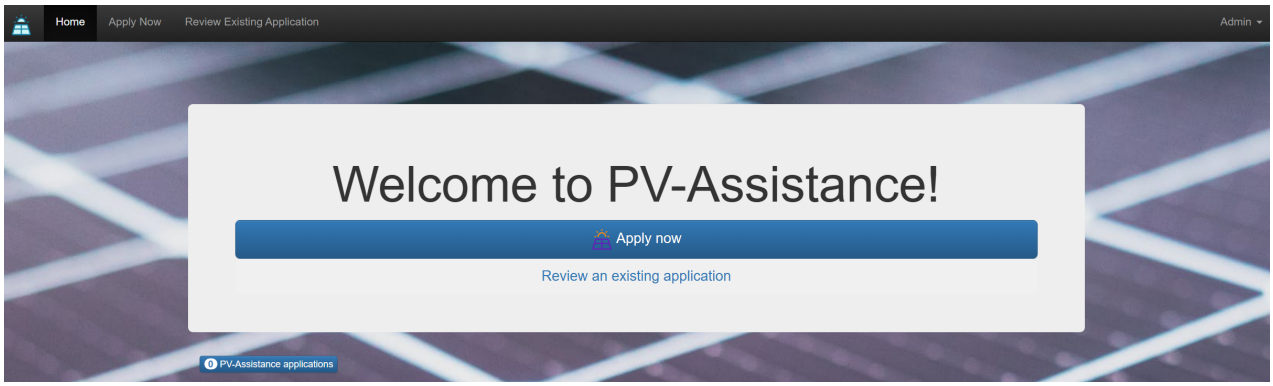
SCR4 PV-Assistance Project, Jack Heseltine

This document can also be found at https://github.com/heseltime/php-myseql-pv-assistance/blob/main/doc/readme.md



Based on php-mysql-bookstore-dev-8 sample project, FHOOe/Hagenberg SCR4 2023.

Test-Login (admin)

user: 'admin'

password: 'admin'

DDEV Instructions

```
ddev import-db --src=db/pv-assistance.sql
ddev describe
ddev start
ddev stop
```

URLs

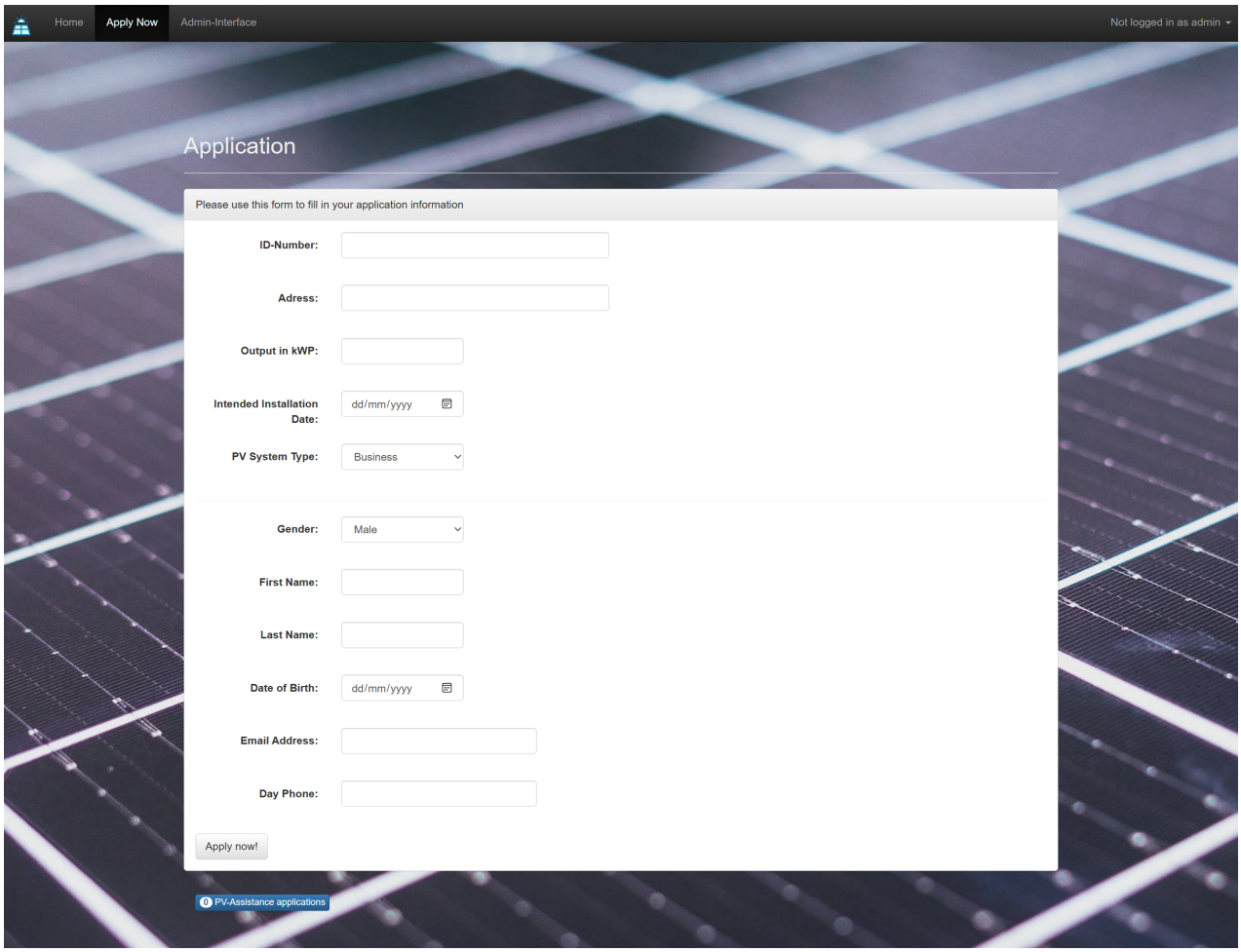
home: https://pv-assistance.ddev.site/

phpmyadmin: https://pv-assistance.ddev.site:8037/

Idea/Project Description and Architecture

Idea

The core architectural idea is a no-credentials log in for the application part, so it is accessible, and a protected area for Sachbearbeiter. The "Sachbearbeiter" (admin) can log in with their credentials and have access to the protected area. The admin can then create, read, update and delete data from the database.

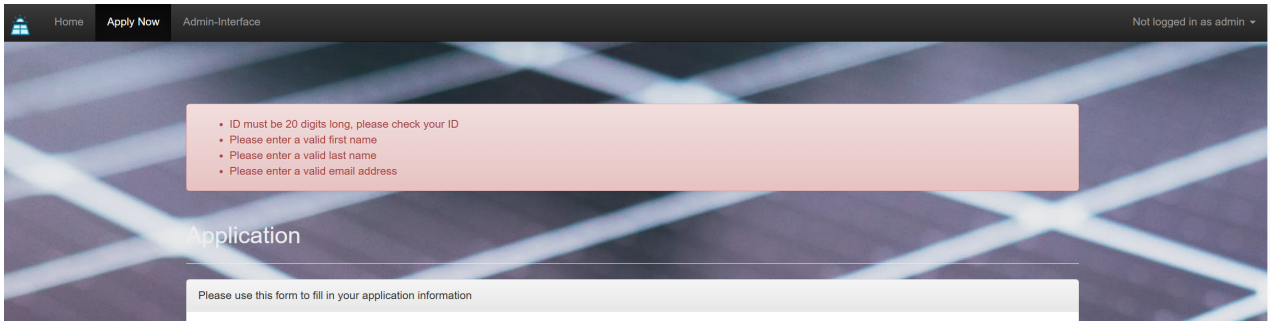


The main application-UI is the form for submission, for admins it is a simple UI allowing status updates and data manipulation (accepted or rejected, optionally notes). There is also a token-protected area for getting the latest version of the application, for the applicant.

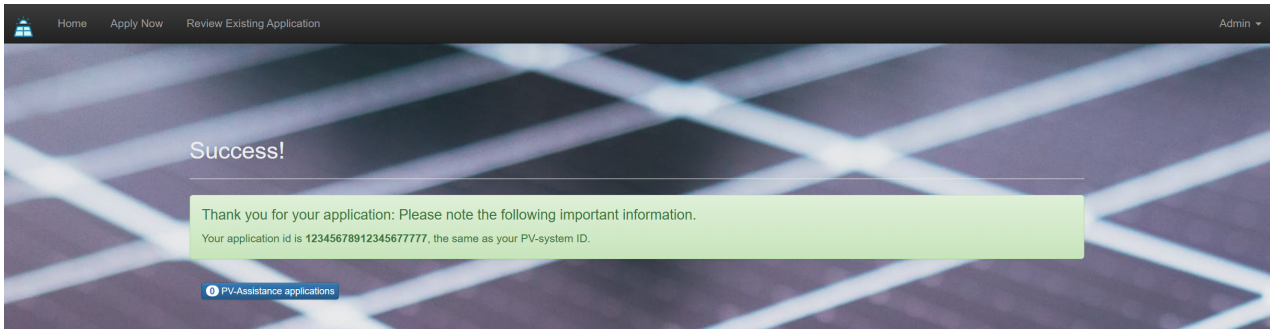
Architecture

The major component is an application form submitted to the Controller (POST-request) to handle input checks server-side, as specified, and either display feedback or a submission confirmation for the application. The DataManager handles writing to the database when all input checks have passed (createUser(), createApplication()): the central element is construction php objects from the query results, and vice versa. These can then also be accessed in the Controller.

Input-Error Case:

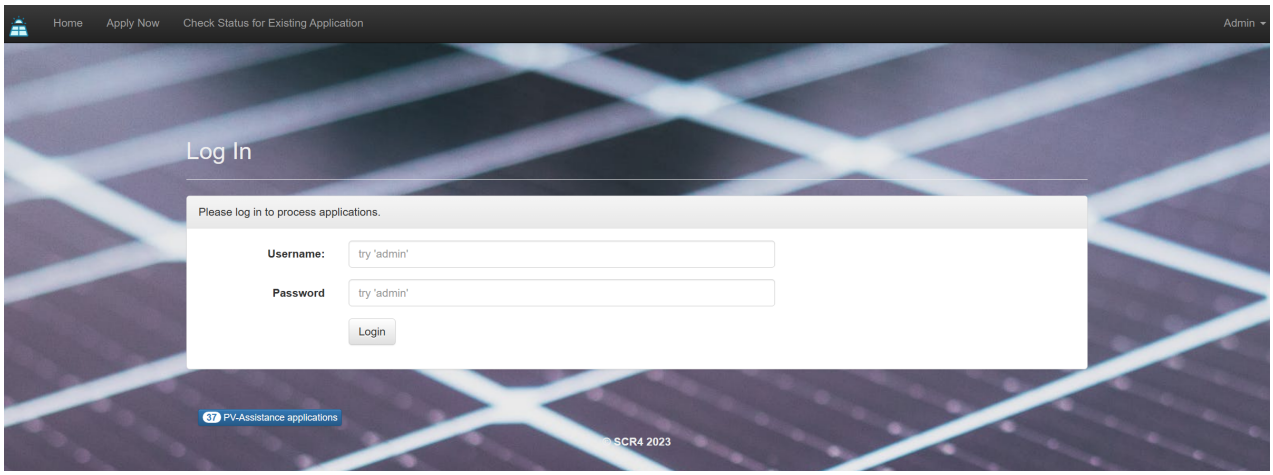


Success-Case:



Everything is also tracked per user and application, especially, including IP address:

Admin log-in:

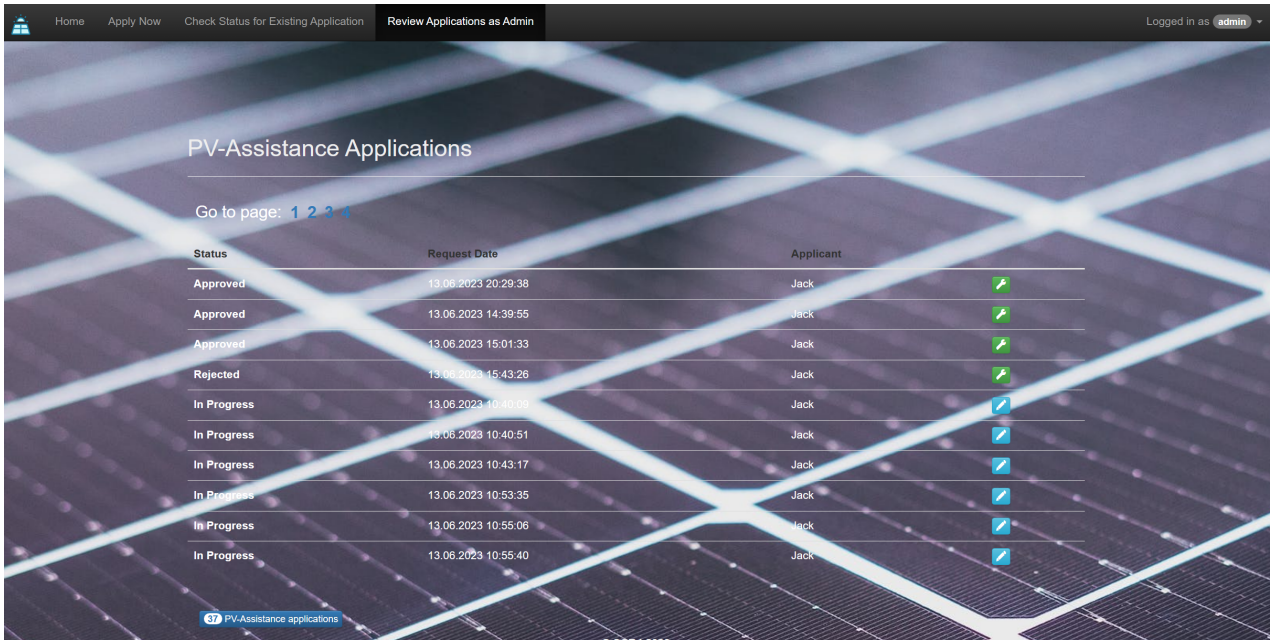


After login an additional header section is available to admins.

Header partial and login actions in the controller/DataManager implement the login functionality for admins. In the logged in state admins have an additional header link that allows them to access the forms for processing. (Admin sign up is out of scope, see existing admin database entries.)

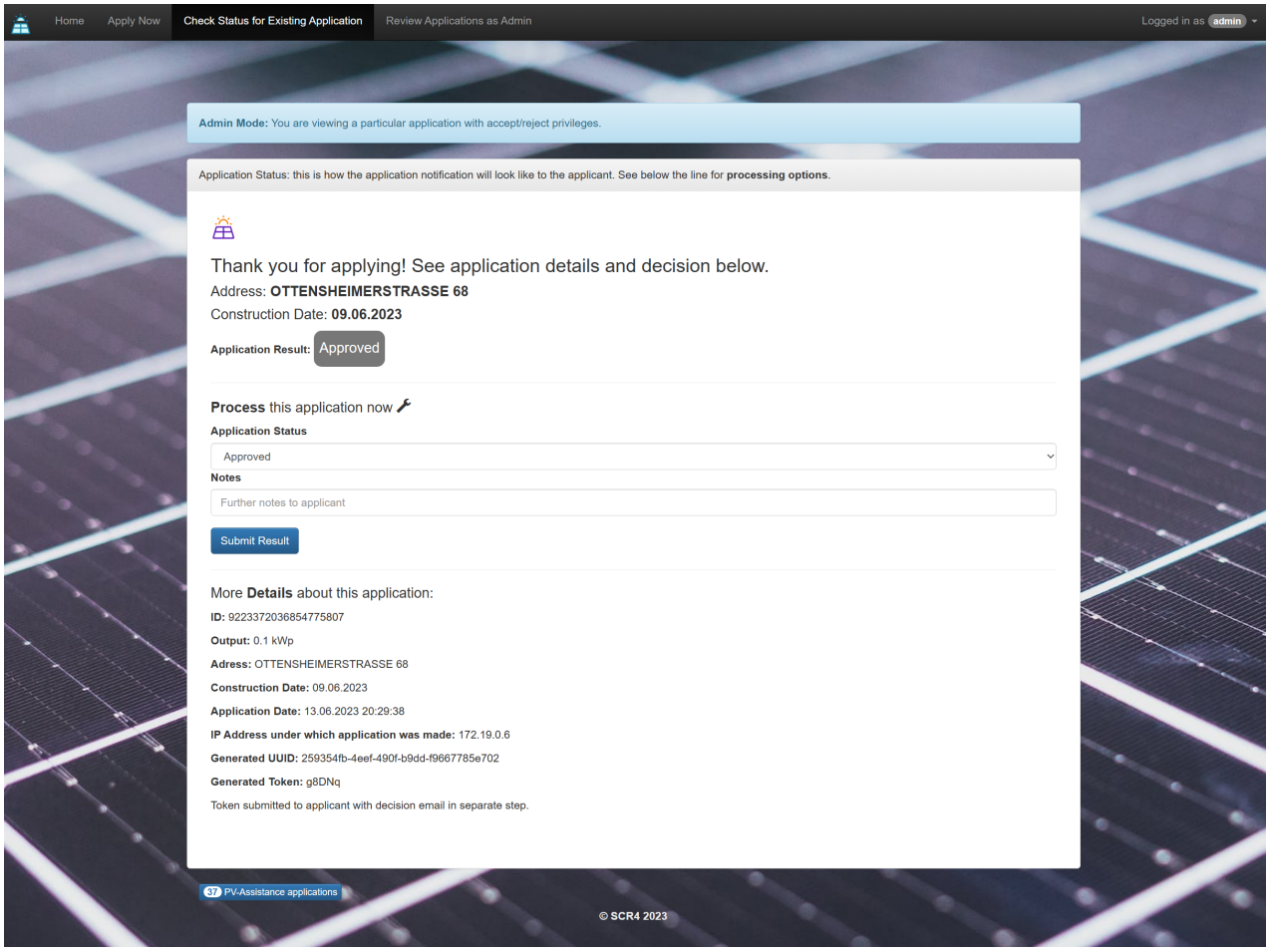
Admin processing:

List view:

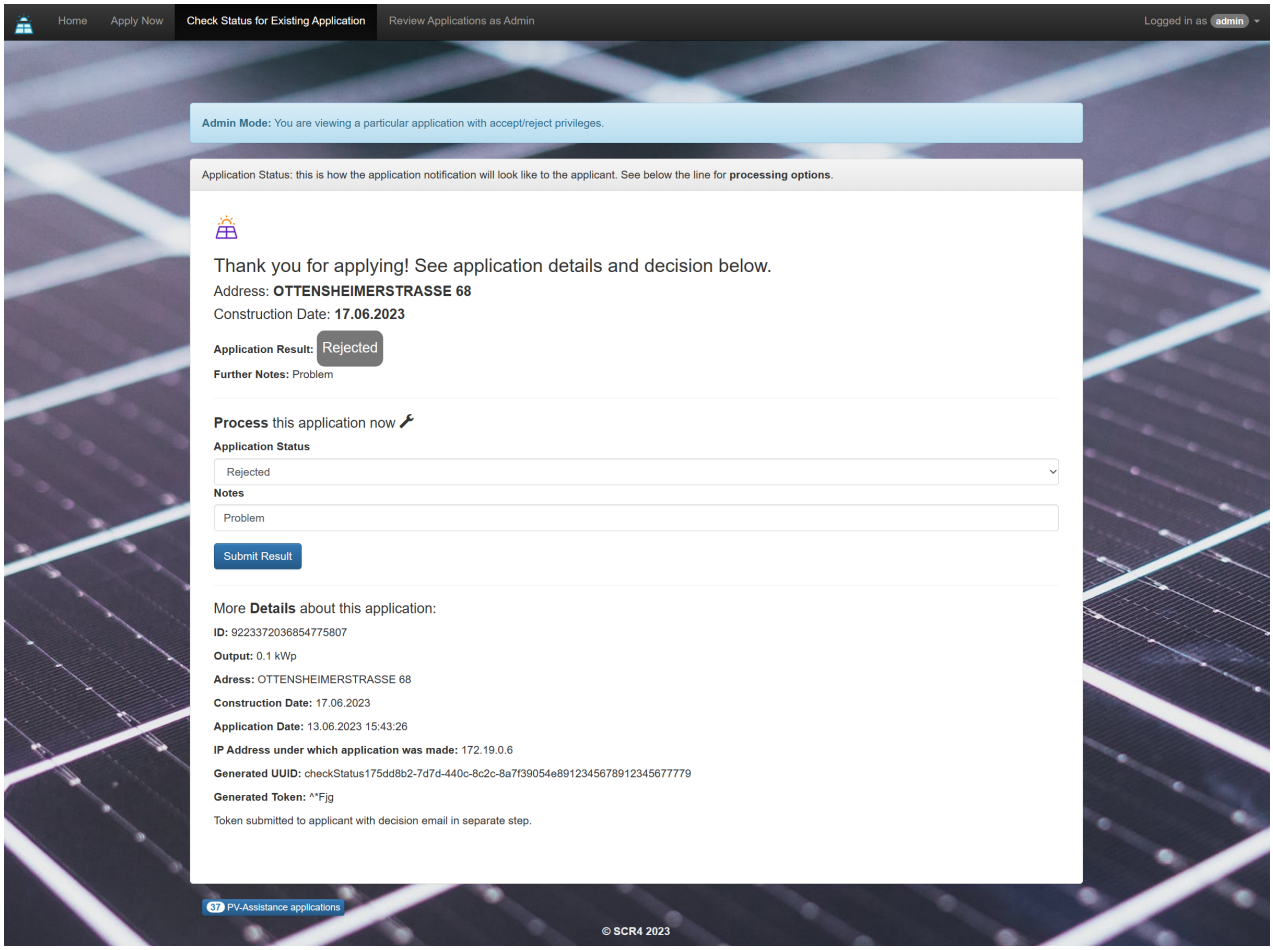


A click on the edit button links to checkStatus tailored to admins, allowing for token processing and similar. The other side of the same view will be the user side.

Approved case:



Rejected Case including note and db representation:

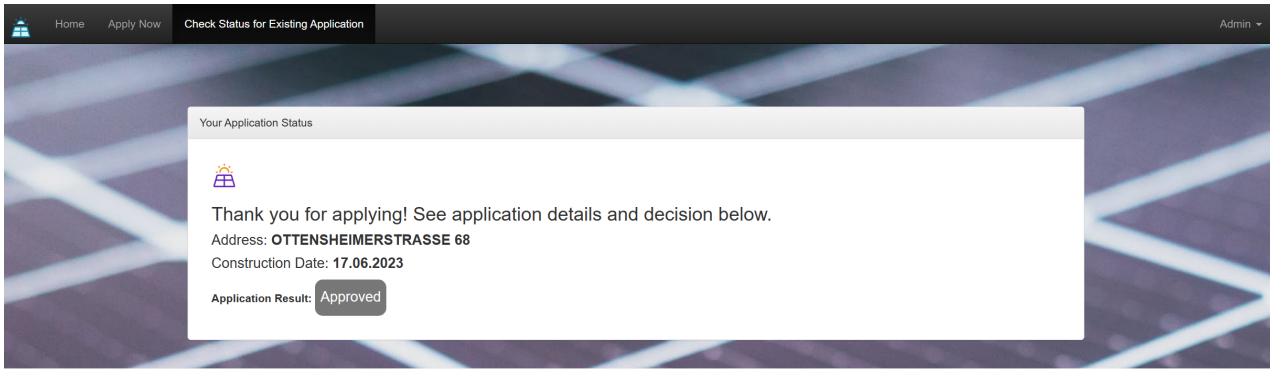


OTTENSHEIMERSTRASSE 68	0.1	2023-06-17	business	2023-06-13 15:43:26	172.19.0.6	^*Fjg	checkStatus175dd8b2-7d7d-440c-8c2c-8a7f39054e89123...	Rejected	Problem

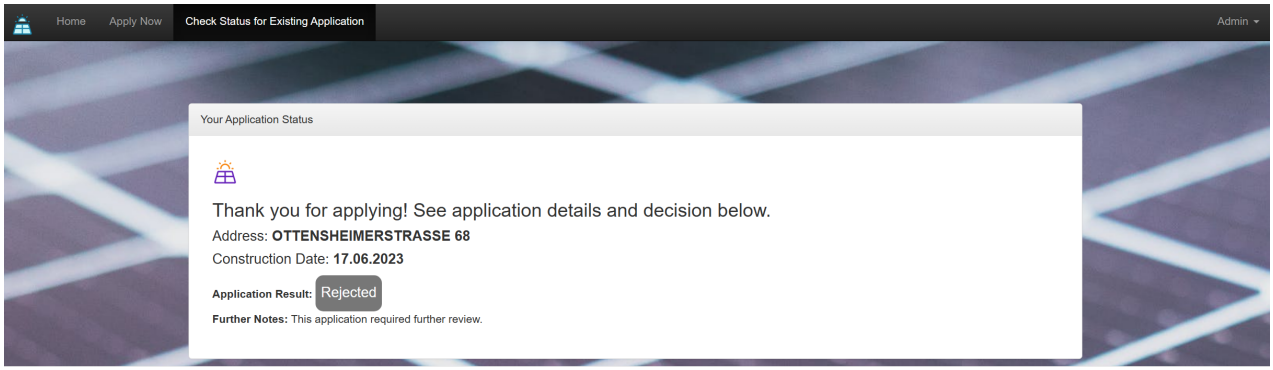
The user is more limited in their options.

Application display once processed (with result):

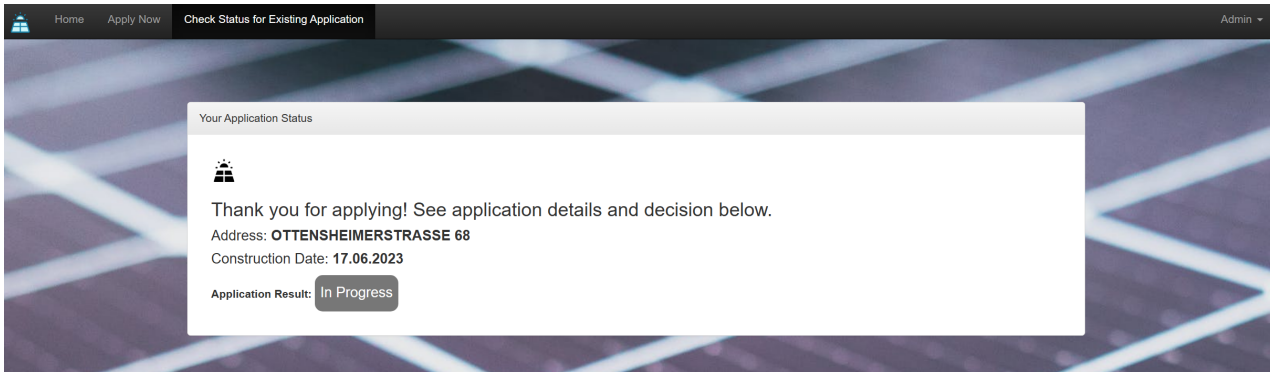
(Approved case:)



(Rejected case:)



For completeness, in progress can also be called and looks like this:



Security: UUID Mechanism

The access URL is encoded in the following snippet.

```
// 32 bits for "time_low"
mt_rand(0, 0xffff), mt_rand(0, 0xffff),

// 16 bits for "time_mid"
mt_rand(0, 0xffff),

// 16 bits for "time_hi_and_version",
// four most significant bits holds version number 4
mt_rand(0, 0x0fff) | 0x4000,

// 16 bits, 8 bits for "clk_seq_hi_res",
// 8 bits for "clk_seq_low",
// two most significant bits holds zero and one for variant DCE1.1
mt_rand(0, 0x3fff) | 0x8000,

// 48 bits for "node"
mt_rand(0, 0xffff), mt_rand(0, 0xffff), mt_rand(0, 0xffff)
```

The code generates a version 4 UUID (Universally Unique Identifier) using random numbers. UUIDs are 128-bit identifiers that are typically used to uniquely identify objects or entities in computer systems.

Here's a breakdown of the code:

The first line generates the "time_low" part of the UUID, which consists of 32 bits. It uses mt_rand(0, 0xffff) twice to generate two random 16-bit numbers.

The second line generates the "time_mid" part of the UUID, which is another 16 bits. It uses mt_rand(0, 0xffff) to generate a random 16-bit number.

The third line generates the "time_hi_and_version" part of the UUID, which is 16 bits as well. It combines a random 16-bit number generated by mt_rand(0, 0x0fff) with the version number 4 (represented by the bitwise OR operation | 0x4000).

The fourth line generates the "clk_seq_hi_res" and "clk_seq_low" parts of the UUID, which together make up 16 bits. It combines a random 14-bit number generated by mt_rand(0, 0x3fff) with the variant bits for DCE1.1 (represented by the bitwise OR operation | 0x8000).

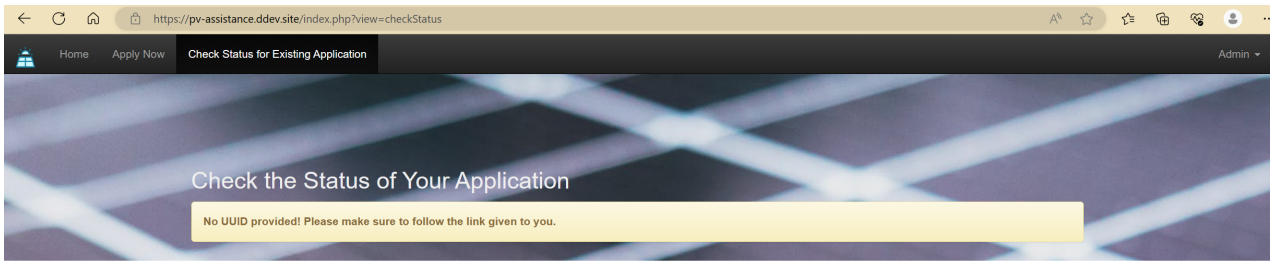
The remaining three lines generate the "node" part of the UUID, which is 48 bits. They use mt_rand(0, 0xffff) three times to generate three random 16-bit numbers.

Overall, the code uses the mt_rand function to generate random numbers within specific ranges and combines them to form a version 4 UUID with the required format and structure.

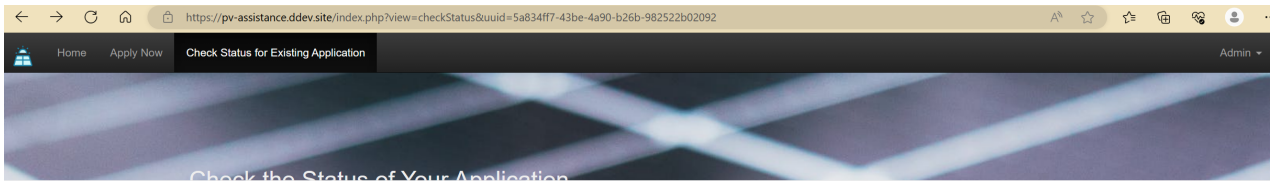
Security: Overall Concept

UUID and token provide two factors for security. The interface is:

(No-UUID case:)



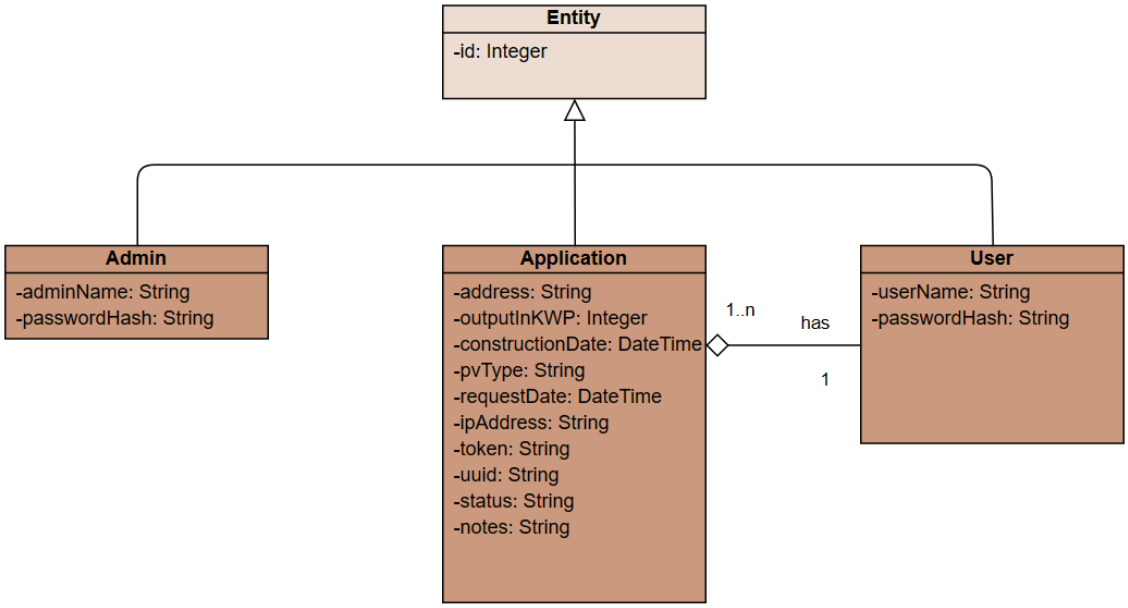
(UUID-link followed correctly:)



As specified, token- and UUID distribution are not implemented here. A simple separation of the two factors when sending (sending at different times) provides a good degree of security.

DB-Structure (UML)

The basic structure reflected in the classes for this project is User for applicants, Admin for application reviewers, Application for applications.



The text-based log is represented in the db, but not in the program class structure.

DB-Structure (SQL)

see PVAssistance.sql

```
CREATE TABLE user (
        userId int(11) NOT NULL AUTO_INCREMENT,
        PRIMARY KEY (userId),
        firstName VARCHAR(50),
    lastName VARCHAR(50),
    sex VARCHAR(10),
    dateOfBirth DATE,
    emailAddress VARCHAR(50),
    phoneNo VARCHAR(20)
  ) ENGINE=InnoDB AUTO_INCREMENT=1 CHARSET=utf8;;

INSERT INTO `user` (`userId`, `firstName`, `lastName`, `sex`, `dateOfBirth`,
`emailAddress`, `phoneNo`) VALUES
(1, 'Jack', 'Heseltine', 'm', '2023-06-08', 'jack.heseltine@gmail.com',
'436649653008');

CREATE TABLE application (
    id NUMERIC(20) NOT NULL,
        PRIMARY KEY (id),
        userId int(11) NOT NULL,
    address VARCHAR(255),
    outputInKWP FLOAT,
```

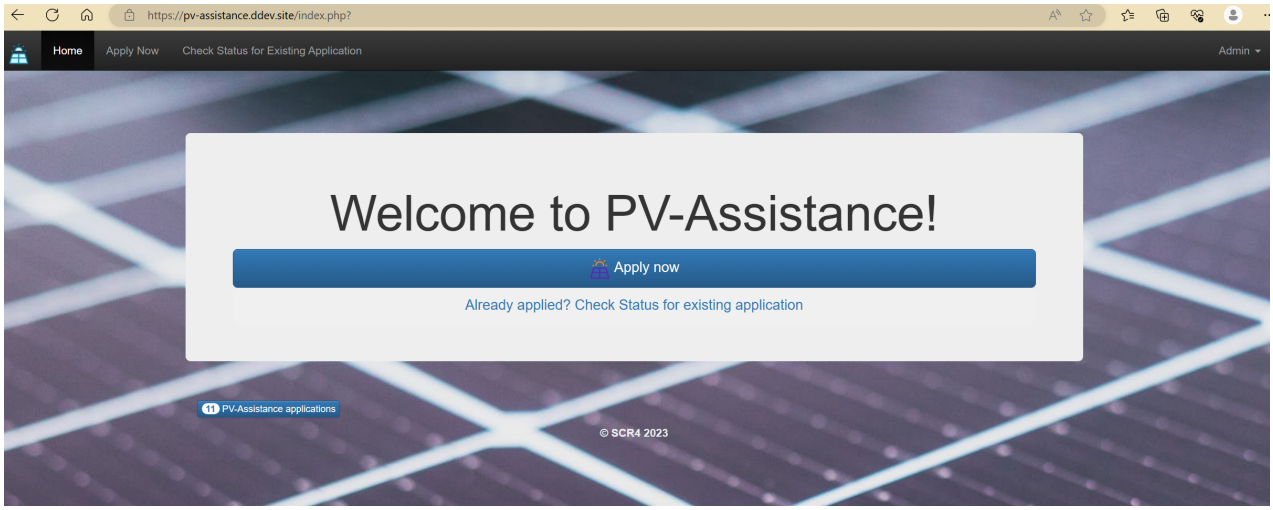
```
    constructionDate DATE,
    PVType VARCHAR(20),
    requestDate DATETIME,
    IPAddress VARCHAR(32),
        token VARCHAR(20),
        uuid VARCHAR(100),
        status ENUM('In Progress', 'Approved', 'Rejected'),
        notes VARCHAR(200),
        KEY userId (userId)
  ) ENGINE=InnoDB CHARSET=utf8;;

INSERT INTO `application` (`id`, `userId`, `address`, `outputInKWP`,
`constructionDate`, `PVType`, `requestDate`, `IPAddress`, `token`, `uuid`, `status`,
`notes`) VALUES
(12345678912345675555, 1, 'OTTENSHEIMERSTRASSE 68', 0.1, '2023-06-09', 'business',
'2023-06-13 20:29:38', '172.19.0.6', 'g8DNq', '259354fb-4eef-490f-b9dd-f9667785e702',
'Approved', ''),
(12345678912345678889, 1, 'OTTENSHEIMERSTRASSE 68', 0.05, '2023-06-17', 'business',
'2023-06-13 17:19:41', '172.19.0.6', 'ZC12M', '5a834ff7-43be-4a90-b26b-982522b02092',
'Rejected', 'This application required further review.'),
(12345678912345678888, 1, 'OTTENSHEIMERSTRASSE 68', 0.1, '2023-06-14', 'business',
'2023-06-13 17:11:41', '172.19.0.6', 'ewTS8', 'checkStatus487c9491-7fdc-4d40-ac20-
4689b61fe197', 'In Progress', ''),
(12345678912345677779, 1, 'OTTENSHEIMERSTRASSE 68', 0.1, '2023-06-17', 'business',
'2023-06-13 15:43:26', '172.19.0.6', '^*Fjg', 'checkStatus175dd8b2-7d7d-440c-8c2c-
8a7f39054e8912345678912345677779', 'Rejected', 'Problem'),
(12345678912345677778, 1, 'OTTENSHEIMERSTRASSE 68', 0.1, '2023-06-17', 'business',
'2023-06-13 15:01:33', '172.19.0.6', 'duJ7!', 'checkStatus4643f918-7b8e-48ac-bf73-
e3c6d7703ef012345678912345677778', 'Approved', ''),
(12345678912345677777, 1, 'OTTENSHEIMERSTRASSE 68', 0.1, '2023-06-17', 'business',
'2023-06-13 14:39:55', '172.19.0.6', 'Lt3*J', 'showRequest73767a4f-c23b-4dca-b9c3-
3cecce3cf92912345678912345677777', 'Approved', ''),
(12345678912345678212, 2, 'OTTENSHEIMERSTRASSE 68', 0.05, '2023-06-08', 'business',
'2023-06-13 12:23:13', '172.19.0.6', 'Yh(WM', 'showRequest0efdbbca-023a-407c-a3c9-
294989eac36a12345678912345678212', 'In Progress', ''),
(12345678912345678211, 2, 'OTTENSHEIMERSTRASSE 68', 0.05, '2023-06-08', 'business',
'2023-06-13 12:14:02', '172.19.0.6', 'G2iBC', 'showRequest0ddcd7e8-d565-4323-b22a-
e3de5e6e225e12345678912345678211', 'In Progress', ''),
(12345678912345678210, 2, 'OTTENSHEIMERSTRASSE 68', 0.05, '2023-06-08', 'business',
'2023-06-13 12:10:50', '172.19.0.6', 'V4@tD', 'showRequest176b7028-d6bf-4f57-ae3e-
b2ef3e5d35a312345678912345678210', 'In Progress', '');

CREATE TABLE admin (
        id int(11) NOT NULL AUTO_INCREMENT,
        name VARCHAR(50) NOT NULL,
        password VARCHAR(50) NOT NULL,
        PRIMARY KEY (id),
        UNIQUE KEY name (name)
  ) ENGINE=InnoDB AUTO_INCREMENT=1 CHARSET=utf8;;

INSERT INTO `admin` VALUES (1, 'admin', '68be59da0cf353ae74ee8db8b005454b515e1a22');

CREATE TABLE log (
        id int(10) NOT NULL AUTO_INCREMENT,
    IPAddress VARCHAR(32),
        action VARCHAR(200),
        userId int(11),
        timestamp DATETIME,
        PRIMARY KEY (id),
```

```
        KEY userId (userId)
    ) ENGINE=InnoDB AUTO_INCREMENT=1 CHARSET=utf8;;
```

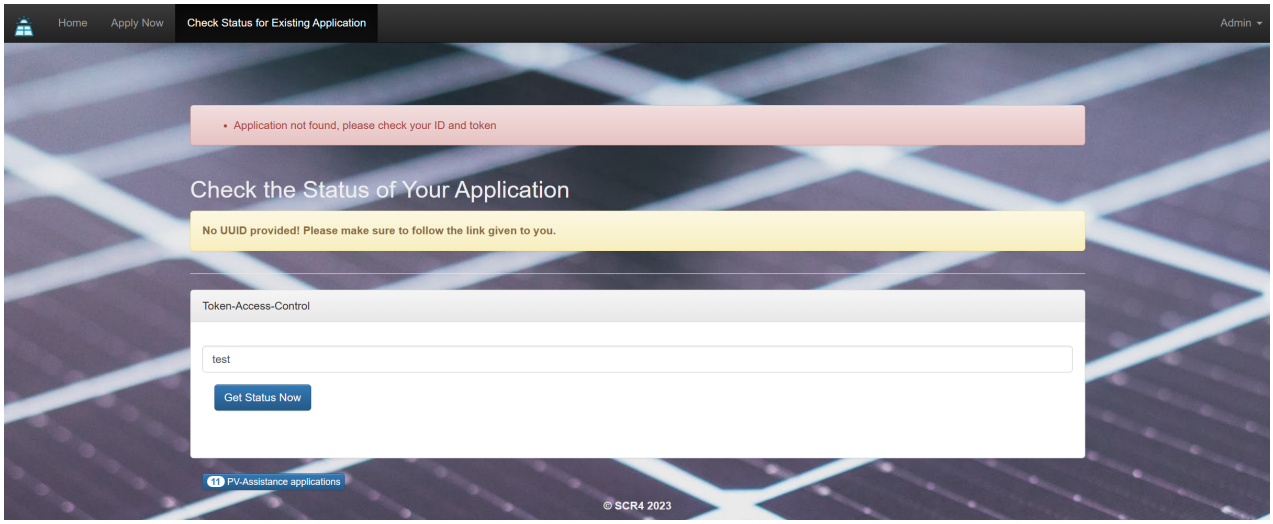
Test Cases

Major functionality tests were already covered in the preceeding Architecture discussion, so I will focus on pen-testing here.

Trying to list applications (by direct link) as non-admin: Redirects to homepage.



Accessing an application without valid UUID has been tested. Valid UUID, but wrong token:



Finally, a log representation, where the logging feature was added at all the sensible junctions.

	id	IPAddress	action	userId	timestamp
Edit Copy Delete	2	172.19.0.6	application with uuid showRequest73767a4f-c23b-4dc...	1	2023-06-15 10:24:36
Edit Copy Delete	3	172.19.0.6	application with uuid checkStatus4643f918-7b8e-48a...	1	2023-06-15 10:33:08
Edit Copy Delete	4	172.19.0.6	new application with uuid 959eb92c-c97a-4449-9678-...	1	2023-06-15 10:50:20