# CREED Experiment Setup Guide

Updated May 2025
*Tell Julian T. if anything is wrong or missing and we can update it!*

## Contents

# 1 Experimental Timeline

## 1.1 Overview

Before running an experiment at the CREED Lab or applying for ACBC funding, present your project at the CREED Lunch Seminar. Contact Julian Kirschner or Madgalena Wasilewska to schedule. Complete the following steps:

1. **Pre-analysis plan:** AsPredicted
2. **Funding application:** ACBC
3. **UvA/EB Project Overview:** Research Management Services
    - Ethics review
    - Data Management Plan

### Pre-Analysis Plan Questions

1. Has any data already been collected?
2. What is the main question or hypothesis?
3. Key dependent variable(s) and measurement method?
4. Number and nature of conditions?
5. Planned analyses?
6. Outlier handling and exclusion criteria?
7. Sample size or rule?
8. Additional info?

# 2 Running Experiments in the Lab

We use the CREED website to manage everything about participant sign-ups. In the lab you run your experiment from your oTree folder on the local server. You need to make sure you i) reserve the lab, ii) get the participants invited, and iii) your code works on all the computers in the lab. Find instructions on how to make sure this all works below.

## 2.1 Reserving the Lab

Coordinate with Joep to reserve lab access and receive credentials for CREEDExperiment.nl.

## 2.2 CREED Website - Managing Participant Sign-up

**Getting Started with the Website.**

- Login details: from Joep

- Experiment number: from Ailko
- URL: `http://www.creedexperiment.nl/proftulp/loginexp.html`
- Create a new session via "new exp"

**Managing Experimental Sessions.**

- Modify session: `Show Session > Modify`
- Send reminder: `Show Session > Reminder`
- Finish session: `Show Session > Finish`

**Inviting Participants.**

1. Create table with name and email.
2. Upload table to invite pool.
3. Prepare a customisable email template.
4. Limit email sends (e.g., 100 per batch).
5. System tracks sent emails and avoids duplicates.

**Sending Reminders.** Navigate to Show Sessions, select a session, and click `Reminder`.

**Tips:**

- You can customise the reminder (subject, message).
- Add your email to test delivery.
- Double-check the session before sending.

### 2.3 Running in the Lab

Here are instructions to implement a super simple workflow when you enter the lab. All you have to do is:

1. turn on all the computers
2. run experiment.bat on the experimenter computer
3. open shortcut 'Chrome to 'experiment' Room Small/Large Lab' on the participant desktop
4. set your session configs in the oTree monitoring page
5. monitor participants in oTree with participant labels of where they are sat

**Configuration of oTree settings.py.** Insert this code into your `settings.py` file. Ensure you don't have other rooms and don't hard-code the ADMIN_USERNAME or ADMIN_PASSWORD elsewhere.

```
ROOMS = [
    dict(
        name='experiment',
        display_name='Experimental Session',
    ),
]

# Admin credentials from environment
import os
ADMIN_USERNAME = os.environ.get('OTREE_ADMIN_USERNAME')
ADMIN_PASSWORD = os.environ.get('OTREE_ADMIN_PASSWORD')

DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.postgresql_psycopg2',
        'NAME': os.environ.get('DB_NAME'),
        'USER': os.environ.get('DB_USER'),
        'PASSWORD': os.environ.get('DB_PASSWORD'),
        'HOST': os.environ.get('DB_HOST', 'localhost'),
        'PORT': os.environ.get('DB_PORT', '5432'),
    }
}
```

Listing 1: settings.py configuration

**Displaying Participant Computer Terminal as Player Labels.** Insert this code into the __init__.py file of your first app.

```
class Player(BasePlayer):
    participant_label = models.StringField()

    def set_participant_label(self):
        self.participant_label = self.participant.label
```

Listing 2: First app init.py

**Start Session Script.** Use this '.bat' script to automate the server and environment start-up:

```
@echo off
REM === Database Setup ===
REM psql -U postgres
REM CREATE DATABASE experimentname;
REM CREATE USER moi WITH PASSWORD '1234';
REM GRANT ALL PRIVILEGES ON DATABASE experimentname TO moi;

REM === Database Management ===
set DB_NAME=experimenting
set DB_USER=moi
set DB_PASSWORD=ExperimentAllDay
set DB_HOST=localhost
set DB_PORT=5432
set DATABASE_URL=postgres://%DB_USER%:%DB_PASSWORD%@%DB_HOST%:%DB_PORT%/%DB_NAME%
```

```
REM === OTree variables ===
set OTREE_ADMIN_USERNAME=admin
set OTREE_ADMIN_PASSWORD=1234
set OTREE_PRODUCTION=1
set OTREE_AUTH_LEVEL=STUDY

REM === Starting the OTree project up on the server ===

REM === !!! ADOPT YOUR PATH !!! ===
cd "C:\Users\Admin\Desktop\TAIT\ExpMistakes Copy"
otree resetdb

REM === !!! SELECT LAB !!! ===

REM SMALL LAB: 145.18.178.133
REM LARGE LAB: 145.18.178.130

start http://145.18.178.133:8000/rooms

echo DON'T FORGET TO SELECT THE RIGHT LAB!!!

otree prodserver

echo Your postgres server is up and running
```

Listing 3: start_experiment.bat

**Monitor the Session.** experiment.bat will automatically open the monitoring page. If not, visit `http://145.18.178.130:8000/rooms` for the large lab or `http://145.18.178.133:8000/rooms` for the small lab.

## 2.4 PostgreSQL Setup

oTree defaults to a basic database called SQLite. This can lag more and does not back up your data in case the computer crashes. Ensure you're using PostgreSQL when running sessions to avoid problems.

If you download your session data after each session and are happy with other wiping the database after every session (but having it stored as a separate file elsewhere, see 3) you can use an existing PostgreSQL database to read and write your data throughout the session (this is already specified to start in experiment.bat). Alternatively, you can also create your own database using i) the code below, or ii) simply opening 'pgAdmin 4' on the experimenter computer and following the steps below.

**Using *pgAdmin 4*.** This is the simplest way with a visual interface.

```
1.  Create a new user and password:
    - Open pgAdmin.
    - Right-click Login/Group Roles $\rightarrow$ Create $\rightarrow$ Login/Group Role.
    - Set:
    - Name: newname
    - Password tab: enter newpassword
    - Privileges tab: check all boxes`

2.  Create the database:
```

```
        - Right-click Databases $\rightarrow$ Create $\rightarrow$ Database
        - Name it experimentname
        - Set owner to newname
```

**Using the terminal.** The master password should be '1234'.

```
psql -U postgres
CREATE DATABASE experimentname;
CREATE USER moi WITH PASSWORD '1234';
GRANT ALL PRIVILEGES ON DATABASE experimentname TO moi;
```

Then set the 'DATABASE_URL' environment variable accordingly in the start_experiment.bat script.

# 3 Managing Lab Session Data

Download it from oTree after each session is completed in the 'data' tab. Save it in some data folder in your project folder as 'Session_XX.csv'. This ensures that whatever happens to the database on the experimenter computer you already have your data elsewhere.

You can then compile this together for your overall dataset.

# 4 Participant Payment

## 4.1 Cash

Ask your supervisor to email Ailko (ailko@xs4all.nl) with:

- Experiment number
- Total cash needed
- Preferred denominations

## 4.2 Digital Transfer

Collect IBANs from participants and forward to central administration.

# 5 FAQ

## 5.1 oTree/Heroku

**Q: How many participants can Heroku handle at once?.** Based on experience, standard setups can support around **20–50 participants** simultaneously. Going beyond this increases the risk of server crashes.

**Q: How can I scale to more participants (e.g., 80–120)?.** A setup that successfully handled up to 120 participants used the following Heroku configuration:

- **Heroku Postgres Standard 0** ($50/month)

- **Heroku Redis Premium 2** ($60/month)
- **Heroku Dynos:**
  - Web dynos: 2X with 4 dynos ($200/month)
  - Worker dynos: 2X with 2 dynos ($100/month)

*Tip:* Dynos and Redis can be scaled up before a session and down afterward to reduce costs. However, the Postgres database must remain active to retain session data.

**Q: Wait pages don't refresh automatically. What should I do?.** If you encounter this issue, make sure to run:

```
heroku config:set PGSSLMODE=require
```

Listing 4: Heroku SSL Configuration

Without this, you may get the error: `FATAL: no pg_hba.conf entry for host`, and wait pages won't auto-update.

**Q: Is there a workaround to prevent stuck wait pages?.** Yes. You can add a fallback script that reloads the page every 10 seconds to avoid session breakdowns:

```
setInterval(function(){ location.reload(); }, 10000);
```

Listing 5: Auto-refresh fallback

**Note:** This helped recover from issues during high-load sessions where the auto-update failed.