1 Server Documentation

This chapter contains documentation for CrowdS server.

1.1 Run Instructions

By following the instructions in this section you should (hopefully) be able to get the current version of CrowdS Server up and running.

1.2 Operating Systems

These instructions have been tried and confirmed working on Windows (v10). They do not however work on macOS (High Sierra v10.13.6), there was a problem with the mysqli connection that PHP does to the MySQL database, further configuration probably needs to be done on the installed PHP Server. CrowdS was previously hosted on a Linux (Ubuntu) system, and should work, but not been tested with this guide. There could be other things that needs to be done to get the system up and running, that I did but forgot to mention in this guide. Its been a while since the time when I first set up the system.

1.2.1 CrowdS Repository

Before you start you have to download CrowdS repository from `https://bitbucket.org/jowalle/crowds/downloads/`. If you can't access this page please contact Mihhail Matskin and ask to be granted permissions.

1.2.2 Software

The software used for this guide is the following:

- PHP
- MySQL
- MySQL Workbench

1.2.3 Install PHP

- Install the latest version of PHP from `http://php.net/downloads.php`.
- Extract the php folder to a place of your choosing.
- Add the path to that folder to your environment variables if necessary.

1.2.4 Test PHP

To confirm if you successfully managed to install the PHP server, we are going to launch CrowdS Admin website by using PHP built in web server.

- Open up a terminal or equivalent for your operating system.
- Change the directory to the html folder inside CrowdS repository. For instance:
  - ***cd PATH/TO/CrowdS/CrowdS Server/PHP/www/html***
- Host the website by running the command:
  - ***php -S localhost:8000***
- If everything worked out you should be able to access `http://localhost:8000` on your browser and see the login screen.
  - Please note that you can not log in before the MySQL system is installed and configured.

1.2.5 Install MySQL and MySQL Workbench

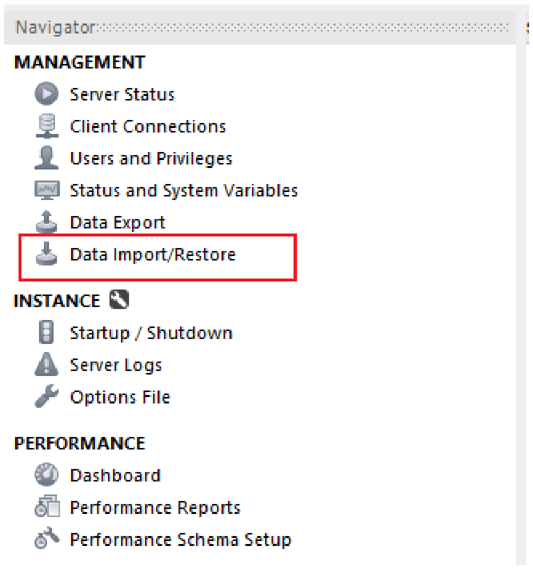
- Install the latest version of both MySQL and MySQL Workbench
  - Use the documentation on `https://www.mysql.com/downloads/` as reference.
  - Depending on MySQL version you might be asked to provide a **password** for the **root** user, do so and make a mental note of it.

1.2.6 Configure MySQL

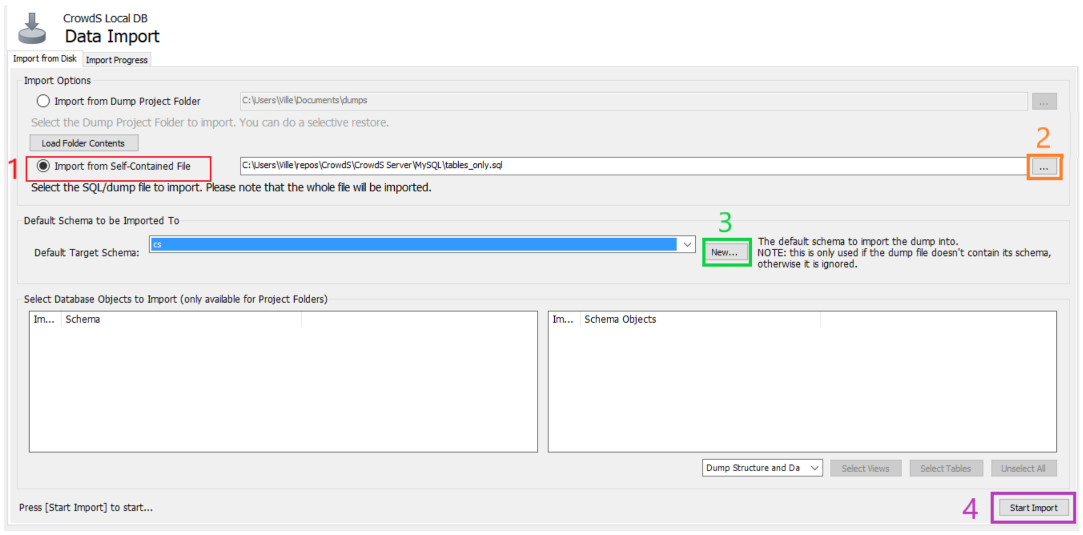
- Modify MySQL Server configuration file
  - The name of the file is **my.ini** or perhaps **my.cnf** (different extensions depending on your operating system).
  - The location might vary, default path for windows is:
    - ***C:/ProgramData/MySQL/MySQL Server 8.0/***
- Add or change the following lines:
  - below [**client**]:
    - **default-character-set = utf8**
  - below [**mysql**]:
    - **default-character-set = utf8**
  - below [**mysqld**]:
    - **collation-server = utf8_unicode_ci**
    - **character-set-server = utf8**

1.2.7 Configure Database

- Start MySQL Workbench
- Open a new connection to the MySQL Server you made while installing.
- Select Data Import/Restore from the Navigator (see image below).



- Import CrowdS SQL tables from datafile (see image below).
  1. Select Import from Self-Contained File
  2. Browse and select the **tables_only** SQL file from the downloaded CrowdS repository (**CrowdS/CrowdS Server/MySQL/tables_only.sql**)
  3. Create a new schema and name it **cs** and use it as Target Schema (the name of the new schema will the name of your database).
  4. Click Start Import



Import from Dump Project Folder: C:\Users\Ville\Documents\dumps
Select the Dump Project Folder to import. You can do a selective restore.

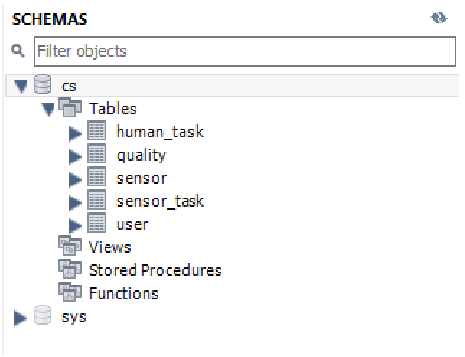
[✓ Import from Self-Contained File]: C:\Users\Ville\repos\CrowdS\CrowdS Server\MySQL\tables_only.sql
Select the SQL/dump file to import. Please note that the whole file will be imported.

Default Schema to be Imported To
Default Target Schema: cs
The default schema to import the dump into.
NOTE: this is only used if the dump file doesn't contain its schema, otherwise it is ignored.

Select Database Objects to Import (only available for Project Folders)

Press [Start Import] to start...

- Switch from the Administration tab to Schemas in the Navigator.
- Check if you can find the newly created schema
  - If not press the refresh button next to SCHEMAS.
  - You should now be able to see all 5 tables (compare with image).



- Finally we have to modify the root user for the database.
  - Currently, php mysqli extension do not support the new **caching_sha2** authentication feature.
  - Make the following query:
    * **ALTER USER '*RootUser*'@'localhost' IDENTIFIED WITH mysql_native_password BY '*RootPassword*';**

        - ∗ Replace ***RootUser*** and ***RootPassword*** with the username and password you picked before.

1.2.8 Initial Configuration of Website

If you followed the guide and got this far, well done! Just one final step, if you remember the website we hosted back when we wanted to test our PHP server, its time to configure it as well.

- Open the file **config.php**
    - Its inside the html folder (**CrowdS/CrowdS Server/PHP/www/html/config.php**)
    - Change the value of **PHP_URL**
    - The value should be the IP address and Port you are hosting the PHP server on.
    - If you want people to reach your server outside the local network, make sure the port forwarding on the router is set up correctly.
- Open the file **mysql.php**
    - Its inside the database folder)
    - Change the value of **username** and **password**

1.3 CrowdS Server API

The following section contains information about all the PHP files that can communicated with by sending a HTTP Post request. The section is divided into 4 different subsections depending on what kind of field each file represents. The purpose of each file is explained, what kind of data it expects as input in the HTTP headers, and finally what kind of encoded message is sent back to the mobile device.

Purpose: The main purpose of the file.

Input: The variables that the file expect to be passed via the HTTP POST method.

Reply: The PHP server will reply a JSON encoded message with a status key which value represents the outcome of calling the file.

1.3.1 Account Files

Files to communicate with when working with user accounts.

register_user.php		
Purpose	Store a new user's info in database	
Input	email	Email of the new user
	username	Username of the new user
	password	Password of the new user
	device_os	Operating system of the user's mobile device
	device_model	Model of the user's mobile device
	firebase	Firebase registration token
	bid	Static bid used in the reversed auction
	device_sensors	Binary representation for sensor availability
Reply	ERROR	User already exist
	OK	User is now registered

delete_user.php		
Purpose	Remove a user's info from the database	
Input	email	Email of the user that needs to be deleted
Reply	ERROR OK	User could not be deleted User is now removed

update_user.php		
Purpose	Update user with new user data	
Input	email password username	Email of the user that needs to be updated Password of the user (could be new) Username of the user (could be new)
Reply	ERROR OK	Could not update user User updated with given data

1.3.2 Session Files

Files to communicate with for user accounts. There is an additional key in the JSON encoded reply message for **ERROR** messages, check the **reason** key for additional information about why the **ERROR** occurred.

login.php		
Purpose	Allow a registered user to access CrowdS	
Input	email password firebase lat lng os version	Email of the user Password of the user Firebase token from the user Current latitude position Current longitude position Operating system of the user's mobile device What version of CrowdS the user is running
Reply	ERROR	Server is currently under maintenance The user is not registered Unknown operating system (has to be android or iOS) The user is running an outdated version of CrowdS
	OK	User has been granted access

logout.php		
Purpose	Marks a user as offline	
Input	email	The email of the user going offline
Reply	OK	User is now considered offline

positions.php		
Purpose	Get the positions of all other active users	
Input	email	Email of the user requesting positions
Reply	OK	Position of all other active users, check the **coords** key

1.3.3 Task Files

Files to communicated with when working with tasks. There is an additional key in the JSON encoded reply message for **ERROR** messages, check the **reason** key for additional information about why the **ERROR** occurred.

create_task.php		
Purpose	Create a task with given data and distribute it to other users	
Input	email	Email of the user that gave the task
	description	A description of what kind of task it is
	lat	Latitude position of task location
	lng	Longitude position of task location
	type	What type of task (hit or sensor)
if type is hit		
Input	hit_type	What kind of HIT
	question	What is the question that needs to be answered
	file	What file has the handles the updates for the task
if type is sensor		
Input	duration	The duration the sensor is read from
	readings	The number of readings that should be done
	sensor	The sensor used
	file	What file has the handles the updates for the task
Reply	ERROR	No devices within the maximum search area
		No device with the required sensor
	OK	Task has been distributed to providers

update_task.php		
Purpose	Store new data for a task in progress	
Input	email	Email of the user that has been given a task
	id	Id for task and provider number for that task (e.g. 568:0)
	data	The new data
	type	Task type
	file	The file that handles updates for the task in question
Reply	OK	Task was updated

ongoing_task.php		
Purpose	Get all tasks that are in progress for the user	
Input	email	Email of the user that needs all ongoing tasks
Reply	OK	An array with all ongoing tasks

task_history.php		
Purpose	Get all completed tasks that has been given by the user	
Input	email	Email of the user that needs all completed tasks
Reply	OK	An sorted array with all completed tasks after completed time

1.3.4 Reward Files

Files to communicate with for data about a users collected rewards for participating.

reward.php		
Purpose	Get the collected rewards for a users participation	
Input	email	Email of the user that needs the reward data
Reply	OK	All rewards collected for participating

1.4 Task Management

I have divided this section into 5 parts, all concerning different kinds of task management. Each part will list all files on the server that it concern and a brief description about their purpose. For more information I recommend studying the source files.

1.4.1 Task Allocation Files

All files are located in the **task_allocation** folder.

- **random:** Selects users by randomly picking users that are above a quality and reputation threshold, if none are found it lowers the threshold and tries again.
- **vcg:** Implements a reversed auction using the VCG mechanism to benefit truthful bidding.

1.4.2 Task Creation Files

All files are located in the **task_creation** folder and handles the creation of the current available types of tasks for CrowdS. Included in the file are functions to notify the user about the newly created task through firebase.

- **multiple_choice:** Creates and stores a multiple choice task.
- **numeric:** Creates and stores a numeric task.
- **sensor:** Creates and stores any sensor task.
- **single_choice:** Creates and stores a single choice task.

1.4.3 Task History Files

All files are located in the **task_history** folder. These files are used by the **task_history** file.

- **hit:** Returns all completed hit tasks for a user.
- **sensor:** Returns all completed sensor tasks for a user.

1.4.4 Task Ongoing Files

All files are located in the **task_ongoing** folder. These files are used by the **ongoing_task** file.

- **hit:** Returns all ongoing hit tasks for a user.
- **sensor:** Returns all ongoing sensor tasks for a user.

1.4.5 Task Update Files

Files are located in the **task_update** folder and the **html** folder. These files are used by the **update_task** file. Included are functions to send a notification through firebase if the task was completed.

- **multiple_choice:** Appends new multiple choice task data to database.
- **numeric:** Appends new numeric task data to database.
- **sensor:** Appends sensor task data to database.
- **single_choice:** Appends single choice task data to database.
- **finish_task:** If a task did not completed within its time frame, this file will terminate it.

1.5 Heartbeat

Heartbeats are scheduled using **at** commands to check if the user are still connected to CrowdS. This is done to remove providers that failed to logout or got disconnected from the pool of active users, to reduce the chance that they are assigned a task. The following files are used to schedule and ping users to see if they still are connected:

- **heartbeat:** Sends an heartbeat through firebase, and schedules a heartbeat check in the near future.
- **heartbeat_check:** Checks if the users replied to the heartbeat, and schedules a new heartbeat.
- **heartbeat_update:** Called by the user to notify the server that it received the heartbeat.

1.6 Quality Control

These files can be found in the **quality_control** folder and are the current available algorithms for the quality control performed by CrowdS.

- **distance:** Calculates quality by a distance formula.
- **majority_voting:** Calculates quality by majority voting.

1.7 Reputation Management

This file can be found in the **reputation** folder and implements CrowdS current simple reputation management.

- **completion_ratio:** Simply calculates user reputation by completion ratio of given tasks.

1.8 Reward Management

This file can be found in the **reward** folder and implements CrowdS current simple reward management.

- **easy_points:** Calculates and updates a users collected rewards.

1.9 Web Page

This section is divided into 5 parts, all connected to the administrative website for CrowdS. Each part has a list of the files related and a brief description about their purpose. As usual, check the source files for more information.

1.9.1 Page Files

Each file in this section is a HTML page, but since they execute some PHP code as well the type had to be of the PHP extension. Every file represents a single web page on the administrator site.

- **index:** The login screen, can only be accessed by administrators. Current way of granting access is by making a query in MySQL Workbench.
- **main:** The web page that is displayed after passing the login screen.

- **account:** A page that has some account management functions, as updating or deleting users.
- **active_users:** Displays all current online users of CrowdS through Google Maps.
- **server_settings:** A settings page which allows an administrator to change settings for the server. For instance what quality algorithm that should be applied. Currently under development and saving the settings is disabled.

1.9.2 Session Files

A session is created and managed while an administrator logs in the website. The following files contains the source code for that process.

- **admin_login:** Creates an session at successful admin login.
- **admin_logout:** Destroys session and changes the header location to login page.
- **forced_logout:** Used to forcefully log out all online users. Usefully if the server has to go down for maintenance.

1.9.3 Settings Files

These files are related to or has important settings that the rest of the system includes.

- **config:** A ton of definitions the server use, a lot of them are paths for the different modules. The idea is that you should be able to change settings dynamically for the different modules, without interfering with the rest of the system. Other noteworthy settings are for number of devices, flag for activating logs and maintenance.
- **session:** Simple short file used for login sessions.
- **settings:** A JSON file containing the settings that can be made on the settings webpage with additional fields for what the latest version is for android and ios.
- **update_server_settings:** Used to update server settings.

1.9.4 Image Files

Images used by the **account** page for a better looking web interface.

- **checkmark-icon:** Check mark image.
- **cogwheel-icon:** Cogwheel image.
- **delete-icon:** Red cross image.

1.9.5 Other Files

A number of files that don't quite fit in under another topic.

- **find_user:** Finds the set of all online users.
- **system_utils:** A helper file with utility functions for other files.
- **info:** Displays PHP info.

1.10 Interface

The following file has all the interfaces that CrowdS Server API requires the rest of the modules to implement. If you would like to design a new module I would recommend implementing the interface for that module as well.

- **interfaces:** This file has interfaces for:
  - Database
  - Task Creation
  - Task Update
  - Ongoing Tasks
  - Task Allocation
  - Quality Control
  - Reputation Management
  - Reward Management

1.11 Firebase

Firebase is used for cross-platform communication with the mobile devices. The mobile devices sends a registration token while creating an user account. To be used for later communication in the future.

- **firebase:** Main file for firebase related things.
- **push:** Used for push notifications.
- **notification:** Used for normal notifications.

1.12 Database

A MySQL database is used for persistent storage. The files PHP files mentioned in this section handles all the queries done by the server.

- **database:** A class that the rest of the system uses for all database related queries. It uses the current select database that implements the database interface.
- **mysql:** This file implements the database interface and are use for all queries.