MILWAUKEE SCHOOL OF ENINGEERING

2016–2017 SENIOR DESIGN PROJECT

Consus Manual

The Four Fifths

advised by
Jon HOPKINS

April 25, 2017

Contents

Chapter 1

Introduction

Our goal was to create an application that will modernize MSOE's Tech Support inventory management system by streamlining the process to check in and out items and providing a way to monitor inventory. Key features of Consus include:

- Barcode and RFID scanners to scan both student IDs and equipment barcodes
- Notifications for students to return equipment
- Tracking all pieces of Tech Support equipment and maintaining information such as manufacturer and a fault history
- Maintaining a list of all equipment checked out by each student
- Tracking all overdue equipment and prevent students with overdue equipment from checking out new items

1.1 The Four Fifths

The Four Fifths is the Software Engineering Senior Design Team that created Consus. The team members are: Quentin Caffero, Tony Fay, Trevin Hofmann, Jordan Longato, and Matthew Mahnke. We were advised by Adjunct Assistant Professor Jon Hopkins during the 2016–17 academic school year.

Chapter 2

Provisioning

This section will describe how to provision a fully functional Consus inventory management system. Consus is comprised of two parts: a server and a client.

Whilst it is possible to run a Consus server and client on "Computer A" and only a Consus client on "Computer B", we do not recommend it. If you want to use multiple check-out stations, we recommend that you have a dedicated Consus server; that is, do not run a Consus server and Consus client on the same computer.

Otherwise, if you are only using one check-out station, you can run the Consus server and client on the same computer.

2.1 Server

The Consus server is where all the magic happens; it performs all the business logic for Consus. The server is required so that it is possible to have multiple client check-out stations working concurrently with the same data. For instructions on provisioning a Consus client, see section 2.2.

2.1.1 Requirements

- Node.js 6, or later
- npm

2.1.2 Installation & Automation

Installation and setup is damn simple for the Consus server. This section will describe how to install the server and setup a task for Windows to automatically start the server whenever the computer starts.

1. Open the Command Prompt and run `npm install -g consus`
2. Open Task Scheduler
3. Click "Create a Basic Task..."
4. Enter an appropriate name and description, such as `Consus server`. Click Next.
5. Select "When the computer starts". Click Next.
6. TODO
7. Enter `consusd start` in the "Program/script" textbox. Click Next.
8. TODO

To start the server immediately, right-click on the task under "Task Status" and select "TODO".

Don't forget to chan
the config file!

2.2 Client

The Consus client is where you as the user get a pretty* view of the data on the Consus server. Since the client doesn't keep any data, a server is also required (see section 2.1).

2.2.1 Requirements

- A Windows 7, or better, computer

2.2.2 Installation

1. Go to `https://git.io/v9YZ0`.
2. Download the Windows build; it will have a name like `Consus-Client-win.zip`.
3. Extract the ZIP archive into a convenient directory, such as, `Desktop` or `C:\Program Files\Consus Client`.
4. Navigate into the `Consus-Client-win32-x64` directory.
5. Right-click on `Consus-Client.exe` and select "Create shortcut".
6. Move the created shortcut to the Desktop, or someplace easily accessible.

The Consus client can now be started by double-clicking the shortcut.

Don't forget to chan
the config file!

*Beauty is subjective, okay?

2.3 Configuration

To ease the maintenance and setup of Consus, you can use configuration files to change some aspects of the software. All config files are located in the `config` directory of each repository's source code.

The file is formatted in Tom's Obvious, Minimal Language (TOML). Refer to https://git.io/vSKUH on how to properly format a TOML file.

The config file loading system is robust. For most purposes if you need to change a variable, you can make that change in the `config/local.toml` file. If the simple `config/local.toml` configuration does not meet your needs, you can refer to https://git.io/vSKUM.

The configuration system can also use your system's environment variables to set values. The mapping of environment variable to config value name is specified in `config/custom-environment-variables.toml`. These environment variable mappings override all other settings from other config files. For more details, refer to https://git.io/vSKTJ.

If you are unsure about anything refer to the config files we have created in `config` directory.

2.3.1 Server

timezone The timezone to represent all times in. Valid time zones are listed under the TZ column at https://en.wikipedia.org/wiki/List_of_tz_database_time_zones.
Default value: "America/Chicago"

Assets

model_photo_directory The location of where the model photos should be stored.
Default value: `assets/img`

Check-in

due_hour The hour equipment is due back, as 24-hour local time.
Default value: 17

due_minute The minute equipment is due back, as 24-hour local time.
Default value: 00

Notifications

Gmail

user The email address for the Gmail account used for sending notification mail.
Default value: `example@gmail.com`, or the `GMAIL_USER` environment variable

password The password for the Gmail account.
Default value: "Pack my box with five dozen liquor jugs", or the `GMAIL_PASSWORD` environment variable

recipient The email address to receive and forward notifications.
Default value: `person@example.com`, or the `EMAIL_RECIPIENT` environment variable

Server

ip The IP address the Consus server should listen on.
Default value: 0.0.0.0

port The port the Consus server should listen on.
Default value: 8000

max_payload_size The maximum payload size received from a client is allowed to be. This may affect the processing of model photos when there is also a large model description.
Default value: 1.5 MB

pid_file The location to save the PID file.
Default value: `consus.pid`

Student

active_status The string that identifies a student as currently enrolled. Consult the document used to import students.
Default value: "C - Current"

2.3.2 Client

timezone The timezone to represent all times in. Valid time zones are listed under the TZ column at https://en.wikipedia.org/wiki/List_of_tz_database_time_zones.
Default value: "America/Chicago"

Assets

max_model_photo_size The maximum size allowed for model photos (kilobytes).
Default value: 950 kB

Cart

timeout The carts timeout. How many seconds should elapse before pending equipment in a student's cart is check out to them.
Default value: 60 seconds

Check-in

due_hour The hour equipment is due back, as 24-hour local time.
Default value: 17

due_minute The minute equipment is due back, as 24-hour local time.
Default value: 00

Server

ip The IP address of the Consus server to connect to.
Default value: localhost

port The port of the Consus server to connect to.
Default value: 8000

protocol The protocol of the Consus server to connect to.
Default value: HTTP

Toast

timeout How many seconds should a toast message be displayed before automatically disappearing.
Default value: 5 seconds

Appendix A

Source Code

All the source code for Consus is open source and freely available at https://github.com/TheFourFifths.

Consus server is an Express server built on Node.js in the JavaScript language. Consus client is a React app built in Electron also in the JavaScript language. Both client and server are implemented using the Flux architecture which permits only a one-way flow of data. Actions are generated from the REST API and passed to the database which keeps persistence of all data, then the action is sent to the dispatcher which dispatches these actions to each data store that subscribes to a particular action type. In the client, the database does not exist because data persistence is maintained by the server. The client also has views that the data stores update when the stores receive relevant actions.