Curtin IET On Campus

Institute of Engineering and Technology

Certificate Maker Kickstart Manual

Harrison G. Outram
Membership Officer

June 26, 2021

The Institution of Engineering and Technology

Contents

List of Figures

1 Introduction

1.1 Context

Curtin IET On Campus (or “CIET” for short) provides multiple industry talks and workshops to STEM students (particularly engineering and computer related science). As part of CIET’s commitment to quality, attendees shall recieve a certificate of attendance stating the name of the event, the club name, the president’s full name with a signature, the the attendee’s full name, and the number of approved CPD hours. This certificate of attendance must be done in a PDF document with vector graphics where possible, then emailed to attendees as soon as possible. This is particularly important for undergraduate engineering students, as one graduation requirement is to obtain a minimum of 16 weighted hours (or five and a third actual) in the PRES category (technical presentations and workshops by a professional body). By creating and sending out these certificates, CIET is holding itself to a high standard of attendee satisfaction and professional development.

1.2 Problem

Despite the necessity of creating and sending certificates, the process for doing so is extremely time consuming and error prone. As shown in Figure 1, this process is comprised of four stages: (1) the attendee record must be collected from the events team and checked for errors, e.g. multiple registrations from one person or missing but required information. (2) one certificate is made for each attendee via the template on CIET’s Canva account. Each certificate is then checked one by one for errors, being recreated if erroneous. (3) the certificates are uploaded to CIET’s MailChimp account, where MailChimp auto-generates a URI for each certificate, required later on. Unfortunately, these URIs involve a randomly generated hexadecimal string, meaning each URI must be recorded manually in the attendee record, then checked for errors. Once the attendee record has been updated with certificate URIs, it is uploaded to MailChimp. (4) The certificates are checked one last time by creating a mock campaign on MailChimp, going through each attendee, downloading the certificate, then checking the certificate for errors and fixing where erroneous. All up, this process takes at least two hours per 50 attendees for someone who has gone through it before, and **far** longer for someone who has not. Worse yet, despite every certificate being checked three times during this process, it is still possible for erroneous certificates to be sent out, which has happened on at least one occasion. Clearly, this process is extremely time consuming and error prone, leading to attendee frustration and a negative impact on CIET’s reputation.

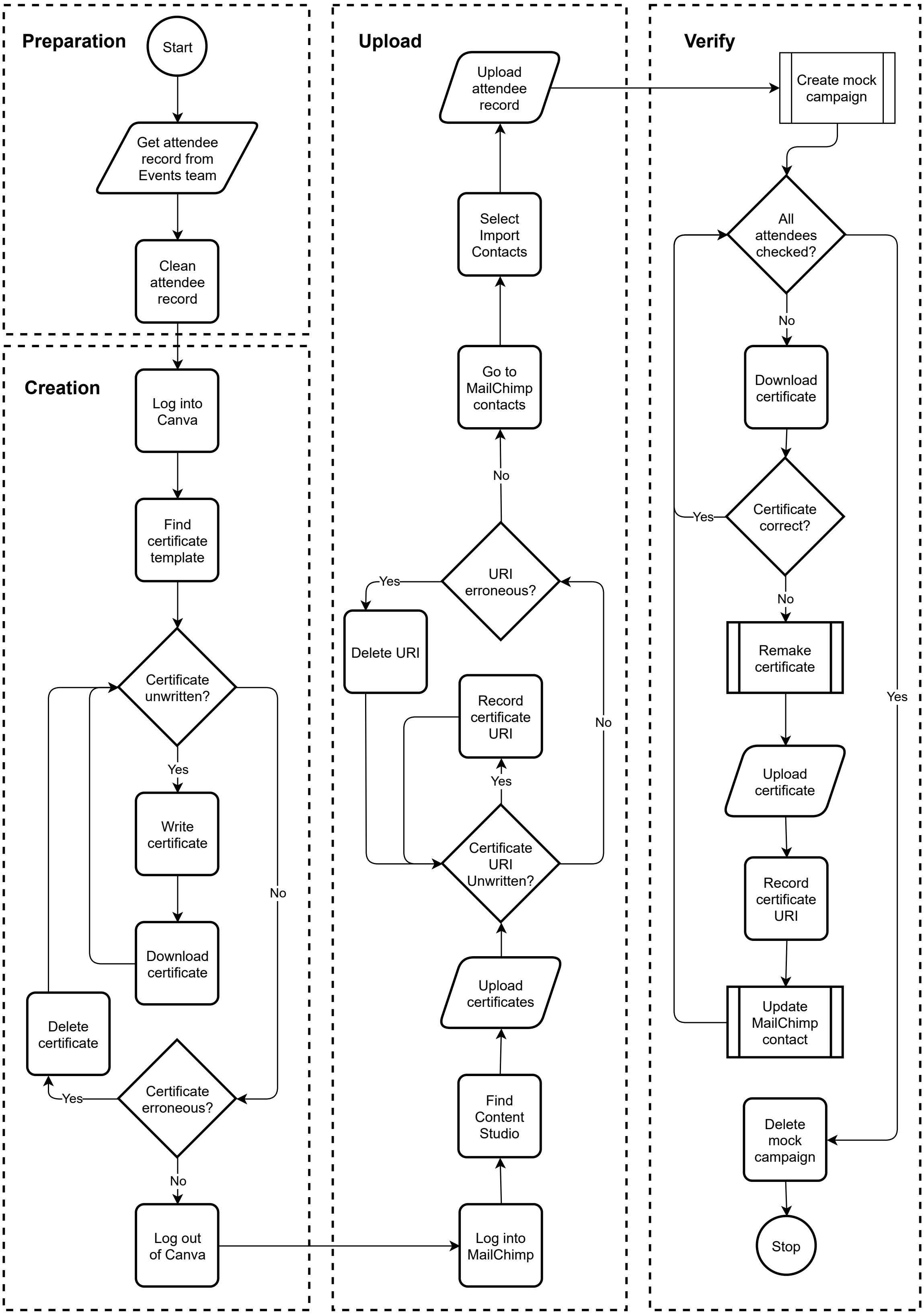

Figure 1: Old solution flowchart.

2 Solution Overview

2.1 Rundown

After some research, it has been discovered that stages 2, 3, and 4 of the old solution can be automated. For stage 1, instead of logging into Canva and creating each certificate manually, the template can be downloaded once and used to automatically create every certificate. The MailChimp API can be used to automatstage 3, where each certificate can be uploaded and have its URI recorded. With the URIs known, the attendee record can be updated and uploaded to MailChimp. Since the certificates were created automatically, stage 4 becomes redundant. This process is sumarised in figure 2. With such a script, CIET can be confident in its ability to deliver the correct certificates to its members on time and without significant human time investments.

Realising this goal, however, will require a well organised team and a plan. Due to the various actions this script must perform, a moderately large codebase utilising several libraries and protocols is needed, demanding a significant time investment and expertise. This complexity is compounded by delegating the workload between a team, needing appropriate version control, task management, and enforeable coding standards. Furthermore, such a project could easily suffer from scope creep, especially with the user interface. Hence, the need for subsequent sections in this document to eliminate these issues.

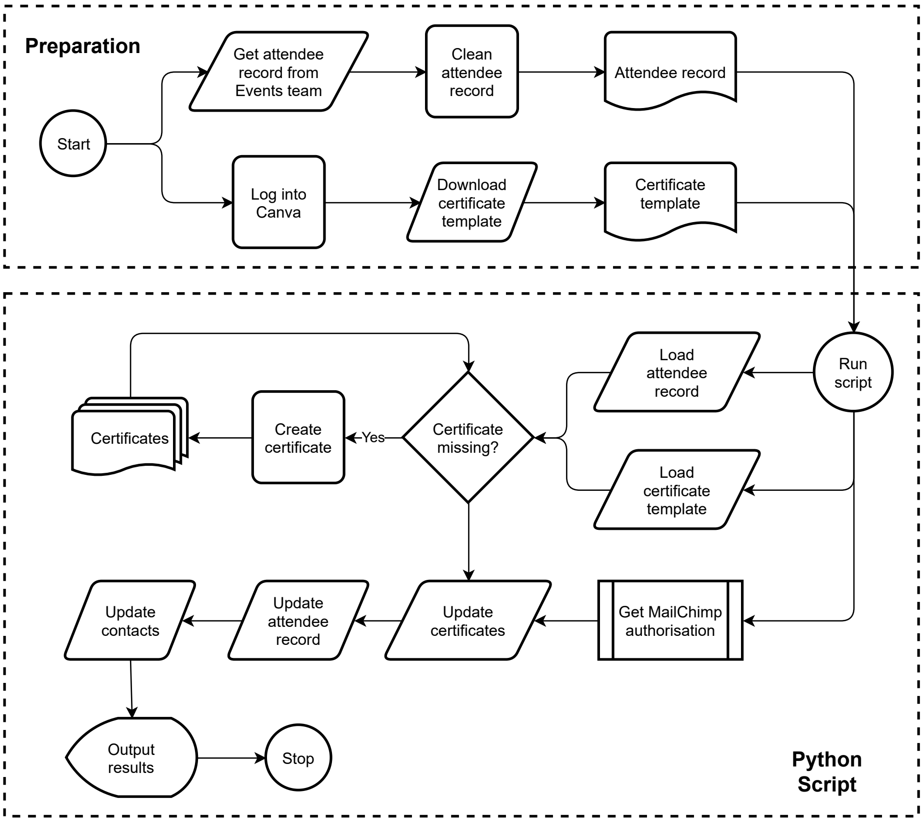

Figure 2: New solution flowchart.

2.2 Prototype

For the sake of time, a prototype can be implemented before scope creep becomes an issue. A prototype must include:

1. Command line interface to run program.
2. Must start program with certificate template path, attendee record path, and Mailchimp API and server keys as command line arguments
   - (a) Strongly recommended to use `argparse` library.
3. Generate a certificate of attendance for each attendee
   - (a) Must be saved into a folder called `/certificates` in the current working directory.
   - (b) Each certificate must be named based on attendee's full name.
4. Upload certificates to Mailchimp, keeping track of the file URIs.
   - (a) Uploading of certificates must be done as a batch, not individually uploaded.
5. Update the attendee record with the certificate URIs.
6. Update the Mailchimp contacts with file URIs.
7. Inform the user of what the program is currently doing.
   - (a) Can just be a simple sentence, e.g. "Uploading certificates to Mailchimp..."
8. Be able to process errors without crashing.
9. Must be able to continue processing certificates when one fails.
10. Must inform user of which step or certificates failed and why.
    - (a) Can be done once the program has finished everything else.
11. Must be able to run on Windows, Linux, and Mac systems capable of running Python 3.

For the sake of time and simplicity, the following assumptions can be made:

1. The certificate of attendance is known.
   - (a) The only field to be changed is the attendee name.
   - (b) The exact location of the attendee name field is known and constant.
   - (c) The placeholder text of the attendee name field is known and constant.
   - (d) The certificate is always valid.
2. The attendee record contains no errors.

2.3 Minimal Viable Product

Once a prototype is complete, a minimal viable product (MVP) can be written. On top of the criteria for the prototytpe, the MVP also has:

1. Can be run via an executable.
   (a) No command line interfacing should be necessary.
2. Should have a simple and initiative GUI.
3. Can select input files via file explorer.
4. Must log at least info, warning, error, and critical messages to a log file.
   (a) Debug messages are optional.
   (b) It must be known where the log file is.
   (c) The GUI should have a button for viewing the log file.
   (d) Strongly recommended to use the `logging` library.
5. Must use OAuth 2 to get Mailchimp authorisation.
   (a) The program should not have the option to use API keys.
6. Should be able to inform the user of errors that occur in real time without interrupting background tasks.
7. Use appropriate colour coding and symbols to inform user of successes, warnings, and errors.
8. Must be able to instantly respond to user inputs without stuttering.
9. Must be able to check attendee record for errors.
   (a) Repeated email addresses.
   (b) Missing names or emails.
10. Must be able to edit PDF document in GUI.
    (a) Must be able to view PDF document within GUI.
    (b) Must be able to select the field where the full name goes.

2.4 Professional Product

If the MVP is done, a professional, market ready, version can be done. This version should have everything the MVP has, and

1. Has ultra-specific error messages.

   (a) Informs the user of what exaclty went wrong.
   (b) Where possible, offers a solution.
   (c) Where possible, offers links to online documentation.

2. Must include install wizard with a build machine.
   (a) Must be able to choose where to install program.
3. Allows user to select where log files go.
4. Allows user to select what types of messages get logged.
5. Must be able to select multiple fields in template as placeholders.
   (a) Must be able to select what each placeholder gets replaced with.
   (b) Must be able to select between attendee attribute, constant value (e.g. event name), or system value (e.g. timestamp).
6. Must be able to generate PDF report of what happened.
   (a) Can select where report gets saved to.
   (b) Can select what goes on the report.

3 Prerequisite Knowledge

3.1 Python Language

3.2 HTTP Requests and Responses

3.3 Application Programming Interface

4 PDF Editing and Creation

5 MailChimp Authorisation

5.1 Using API and Server Keys

5.2 Using OAuth 2.0

6 Certificate Uploading

7 Contact Updating

8 User Interface

8.1 Command Line Interface

8.2 Desktop GUI

9 Version Control

9.1 What is Git?

9.2 Using Github

9.3 Recording Library Requirements

9.4 Virtual Environments

10 Coding Standards

10.1 Readability and Maintainability

10.2 Documentating using Sphinx