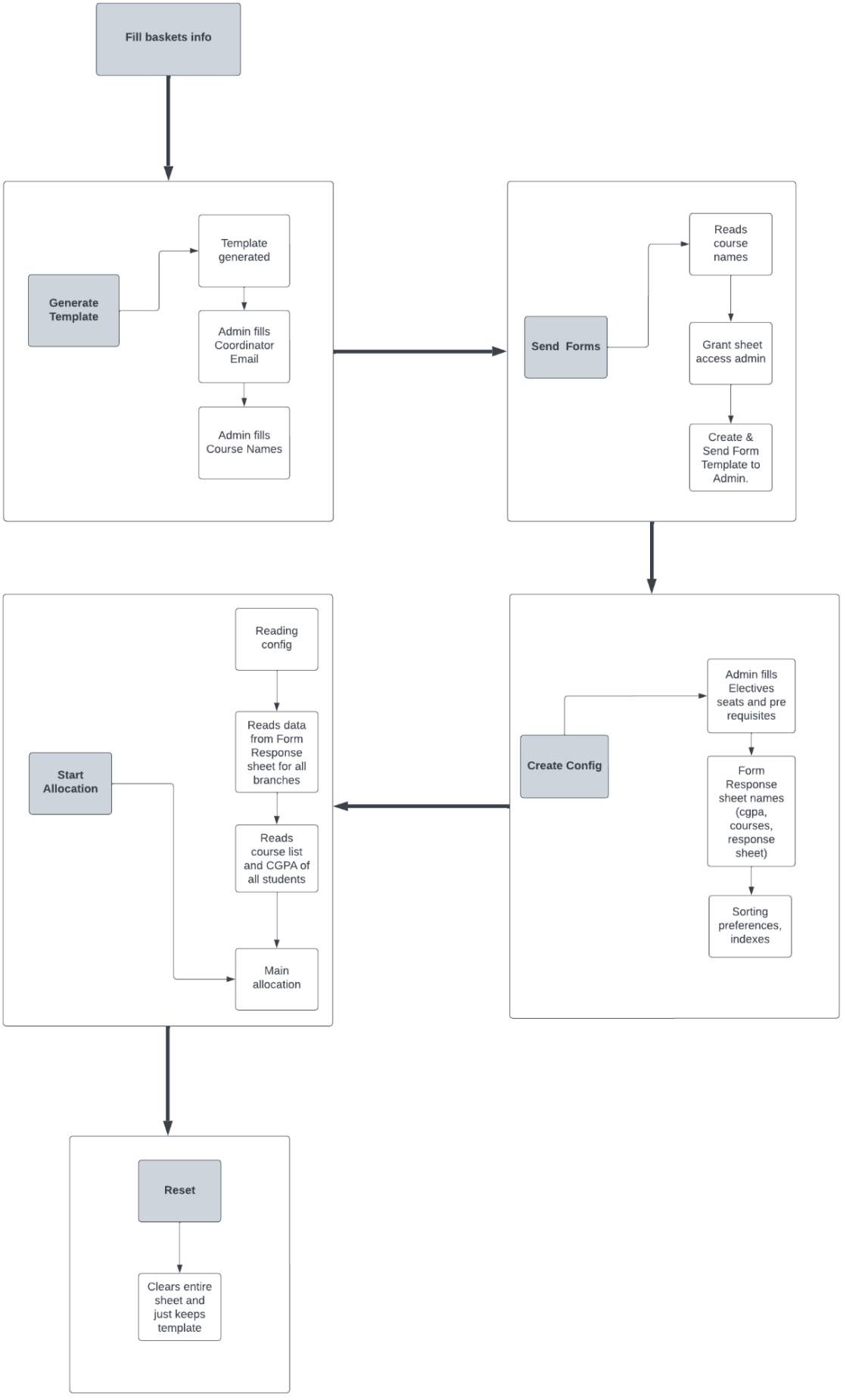
Student Elective Course Allocation Report

Table of Contents

1 High Level Design

Introduction

The Student Elective Course Selection Management system aims to simplify the task of handling student elective course management within Excel using Google Apps Script. It serves as a valuable tool for administrators, streamlining the allocation of elective courses to students efficiently. In this report, we will explore how this system operates for elective course allocation.


Fill baskets info
Generate Template
Template generated
Admin fills Coordinator Email
Admin fills Course Names
Send Forms
Reads course names
Grant sheet access admin
Create & Send Form Template to Admin.
Create Config
Admin fills Electives seats and pre requisites
Form Response sheet names (cgpa, courses, response sheet)
Sorting preferences, indexes
Start Allocation
Reading config
Reads data from Form Response sheet for all branches
Reads course list and CGPA of all students
Main allocation
Reset
Clears entire sheet and just keeps template

There are five Functions that will be used in this project and a brief overview of each function is given below

1. **Generate Template:** First, the administrator has to fill in his email and the details for a number of courses and baskets. Later upon successful activation of the 'generate template' trigger a template for course input is created where the administrator has to fill in the names of the courses.

2. **Send Forms:** This trigger uses the Google Form API to create a Google form with student details and a grid of courses for preference selection. Once the form is created it is sent to the administrator where he/she can add additional fields like declaration, CGPA etc.

3. **Create Config:** This function is integrated with the above trigger and it is not called explicitly. This function takes all the required parameters like course prerequisites, number of seats, etc which will be used for allocation and further functions

4. **Start Allocation:** This function takes the students' responses along with the parameters from the config sheet and starts running the algorithm for allocating the electives to respective students based on the specified preference

5. **Reset:** This function clears all data and deletes the additional sheets apart from the template sheet. This function needs to be called once the allocation process is completed and all the response sheets are downloaded. This function can also be used in case of errors while generating the template but prevent it from being it once the preference form is sent to the students.

2 Low level Design

2.1 Generate Template

This functionality has been designed to capture essential initial information related to elective courses. This information includes details such as the number of baskets per branch and the quantity of courses within each basket, along with their respective names. Now, let's explore the functions employed within this module:

1. **generate_for_branch()**: This function requires two parameters, "data" and "col_num," and is intended to populate a spreadsheet in a specific format. It initiates from a specific row (row 4) within the data and parses a JSON string located in the designated column (specified by "col_num + 1"). The expected JSON format represents a list of positive integers, and the function proceeds to update the data, creating entries for "Basket" and "Course." It performs checks to ensure the input's validity and raises appropriate errors if the input does not adhere to the expected format or if the list contains non-positive integers. The function then returns the modified data.

2. **generate_template():** This function acts as the starting point for generating a spreadsheet template. Initially, it verifies the integrity of the existing data and confirms that the sheet has a predefined size of 5 rows and 10 columns. It subsequently defines the maximum number of rows and columns for the template. Subsequently, it invokes the "generate_for_branch" function three times with distinct column offsets (0, 4, and 8), effectively populating the spreadsheet with data for three different branches. Finally, it writes the generated data to the sheet. In case any errors arise during the process, the function will throw an error message to indicate the issue.

3. **write_data() :** This function accepts two parameters: the first is the sheet name, and the second is the data. Its purpose is to write the provided data into the specified sheet.

2.2 Send Forms

Send Forms module is used to generate the required elective preference Google form which the students will fill out. This module is designed to automate the process of creating a preference form for each branch. A detailed explanation of multiple sub-functions is mentioned below:

1. **get_courses():** This function fetches the courses list for each branch from the template sheet. It is an important function as it will be reused whenever courses are required.

2. **createMCQ():** This function is used to create a multiple-choice grid within a Google Form. It creates a grid item in the form with preferences (Preference 1, Preference 2, etc.) as rows and subject names as columns. The important condition here used while creating the grid is that a subject can have only one preference.

3. **send_forms():** This is the main function that logs the electives for different branches (CSE, ECE, DSAI) by calling the get_courses() function and passes this information to createAndSendForm() which creates the appropriate fields in the form.

4. **createAndSendForm():** This function creates a Google Form using Google Form API for collecting elective preferences from students. It takes the administrator's email, a list of subjects, and the subject name as input. It creates a form with some mandatory fields like the student's name, the student's roll number and the subject preference grid.

Once the form is created successfully, it will be sent to the administrator email where he/she has access to edit the form, adding

additional fields along with the mandatory fields. Once the form is finalized it can be sent to the student to collect their response

2.3 Create Config

This module serves the purpose of gathering user input data and configuring various settings for the allocation process. To facilitate this functionality, a range of functions is employed, including:

1. **config():** This function is responsible for creating the configuration sheet, as well as sheets for courses and the Cumulative Performance Index (CPI).

2. **check_if_sheet_exists():** This function checks for the existence of a sheet to avoid creating it redundantly.

3. **write_data():** This function is utilized to record the necessary data in the respective sheets.

Within this section, users interact with different types of sheets, which include:

1. **Config Sheet:** This sheet contains all the essential configurations required for course allocation. It encompasses all the parameters necessary for subsequent functions.

2. **CGPA Sheet:** The CGPA sheet serves as a repository for collecting students' CPI inputs

3. **Course Sheet:** In the course sheets, users provide essential information about courses that students have previously completed.

2.4 Start Allocation

The Start Allocation module serves as the central component of our software, orchestrating the allocation process. In this section, we will delve into the underlying mechanics of the various functions within this functionality.

1. **get_preference_list():** This function retrieves the customized preference list established by the user, which is a critical component of the allocation process.

2. **get_map_electives_count():** This function provides a mapping of elective courses to the number of available seats for each elective. For instance, get_map_electives_count()["Ethics"] reveals the current seat availability for the "Ethics" elective at any given moment.

3. **get_map_electives_requisites():** This function furnishes a mapping of elective courses to their respective prerequisite courses. For instance, get_map_electives_requisites()["Ethics"] offers a list of prerequisite courses required for the "Ethics" elective. This information is vital for verifying students' eligibility for elective courses during allocation.

4. **get_map_cgpa():** This function produces a mapping that associates students' registration numbers with their respective Cumulative Performance Index (CPI) values. For example, get_map_cgpa()['20bcs120'] returns the current CPI of the student with registration number '20bcs120'.

5. **get_map_courses():** This function generates a mapping that links students' registration numbers to a list of all the courses they have enrolled in and successfully completed. For instance, get_map_courses()['20bcs120'] provides a list of courses that '20bcs120' has enrolled in and passed up to the current date. This information is instrumental in assessing students' eligibility for elective courses during allocation.

6. **<u>get_responses()</u>:** This function amalgamates responses from all branches, organizing them into a 3D array. Each row in this array represents a single response which is a 2D array. The first row of each response contains response information, including the timestamp, CGPA, registration number, and the student's branch. Subsequent rows of each response contain the student's preferences for each basket, ranked from highest to lowest preference. For instance, a response for '20bcs120' might appear as follows:

```
[
    [ 31921.31 , 9.8 , '20bcs120' , 'cse' ] ,

    [ 'Computer Networks' , 'Data Structures' ] ,

    [ 'Linear Algebra' , 'Probability Theory' , 'Discrete Math' ]
]
```

According to this response, for students with reg no ‘20bcs120’, the first preference in the first basket is ‘Computer Networks’ and the second preference is ‘Data Structures’. Similarly in the second basket, the first preference is ‘Linear Algebra’ and ‘Probability Theory‘ & ‘Discrete Math’ are second and third respectively.

7. **<u>start_allocation()</u>:** This is the core function within the module. It orchestrates the entire allocation process by calling the functions mentioned above. The allocation algorithm consists of the following steps

Step1 :

Sort the responses based on the preference list. If the preference list is [0,1], it means sorting first by time and then by CPI; for the preference list [1,0], the order is reversed.

Step 2:

Progress to the next response, and if you are already at the end, proceed to step 8.

Step 3:

Move to the next basket within the current response. Return to step 2 if already at the end.

Step 4:

Advance to the next course within the current basket. If already at the end, terminate the algorithm due to insufficient available seats.

Step 5:

Check if seats are available for the current course; if not, return to step 4.

Step 6:

Verify whether the student has enrolled in and passed all the prerequisites for the course. If the prerequisites are met, proceed to step 7; otherwise, return to step 4.

Step 7:

Allocate the course within the current basket to the student, and decrease the number of available seats for that course by 1. Return to step 3.

Step 8:

The allocation process is complete.

8. **organize_data_by_branch():** This function is responsible for organizing and presenting the allocation results categorized by branch into new sheets, making it easier to visualize and understand the outcomes.

2.5 Reset

This module serves the purpose of erasing all data within an Excel sheet following the completion of course allocation and once the administrator has downloaded all the sheets. It comprises two key tasks:

Clearing the template sheet: Initially, this function clears the template sheet that was filled out by the administrator at the start of the process. This ensures that the sheet is devoid of any information.

Deleting intermediate sheets: The function also removes any additional sheets that were generated during the intermediate steps of the process, providing a clean slate for future use.

After resetting the Excel sheet, the function proceeds to create a new initial template. This reset option can also be utilized in case any errors occur during the template generation, allowing for a fresh start with a clean sheet.

3 Error Handling

3.1 Generate Template:

- **generate_template() -> creates template**
  - ➢ Template page is corrupted (check data object length and width)-> Reset the page
- **get_template_for_branch(data_object,col_name) -> Writes template to object for given branch**

- ➢ If Bucket-Courses input is in wrong format -> Refer to docs
- ➢ If Bucket-Courses input has zeros -> zeros can't be an input
- ➢ If Bucket-Courses input is not int-> Input must be int

3.2 Send Forms :

- ❖ **send_forms() -> sends forms administrator access to administrator mails**
  - ➢ Email wrong -> (raise an error if admin error is wrong)
- ❖ **get_courses(row,col,word) -> Returns list of buckets and courses in them (2d array)**
  - ➢ Raise error if course name is null -> (Error : (row, col) is empty)
  - ➢ Check course duplications
  - ➢ Verify the number of branches & number of courses in each basket -> (Error: where the error is)
- ❖ **createAndSendForm(email, subjectsList, subjName) -> creates and sends the administrator access to the form**
- ❖ **createMCQ(form, subjectsList) -> Creates preference-elective mapping input in form**

3.3 Config:

- **config() -> creates a sheet with all the courses and their prerequisites along with some other details like ECE Form response sheet name, CSE Form response sheet name etc**
  - First we delete all the sheets except the sheet1 and -> then create the config sheet along with the cgpa and the course sheets**.**
  - If the administrator does not give any input in the basket courses input -> then we do not generate the cgpa and the course sheet forms for that branch.

3.4 Start Allocation:

- **get_config_sheet_details() -> get all required details from the config sheet**
  - If the admin does not fill the timestamp index/reg no index/preference start index -> throw an error.

- **get_responses() -> get all branches form responses**
  - If the branch has electives but no form response sheet name is filled in config sheet -> throw an error.
  - If reg no is not present in map_cgpa -> This means that regno is not there in the cgpa sheet which raises an error stating the reg no and issue.

- **get_branch_response() -> get all form responses of a branch**
  - If admin has added/removed courses directly in the form and changes are not reflected in the template -> check courses and compare with the courses list from get_courses(); if there is any inequality, then throw an error to admin to update the course list in the template
  - Check out of bounds

- **start_allocation() -> main function which allocates the electives**
    - If none of the courses in a basket gets allocated -> then it throws an error to increase seats for subsequent branches and the basket no.
    - If the course is not there in get_map_electives_requisites() -> then it throws an error to add the course.

- **get_map_courses -> returns the object having the registration number and the courses**
    - If the given sheet name is wrong-> then it will throw an error stating “sheet name not found”
    - If the electives does not exist for some branches-> then the sheets related to those branches are not created and hence the data related to that particular branch is not included in the object which is being returned.
    - If reg no is not present in map_courses -> This means that regno is not there in the courses sheet and map takes an empty list as default value.

- **get_map_cgpa -> returns the object having the registration number and the cgpa**
    - If the electives does not exist for some branches-> then the sheets related to those branches are not created and hence the data related to that particular branch is not included in the object which is being returned.
    - If the cgpa is not in float datatype then the function will throw an error.

- **`create_new_sheet` -> creates new separate sheets to keep the data generated, in their respective branch sheets**

    - ➢ If the sheet already exists -> then we'll just clear the sheet data and return the sheet to the caller function
    - ➢ If the sheet doesnot exist -> then we'll create a new sheet and return that sheet to the caller function
- **get_map_electives_count() -> get a map of courses to number of seats**
    - ➢ If the entered number of seats is not an Integer -> throw an error.
    - ➢ If the entered number of seats value is not filled -> throw an error to fill all the required details

- **get_map_electives_requisites() -> get a map of courses to its prerequisites**
    - ➢ If the entered prerequisites are not in a list form-> throw an error.
    - ➢ If the entered courses is not filled -> throw an error to fill all the required details

- **get_preference_list() -> gets the preference for sorting (CGPA or FCFS)**
    - ➢ If the entered value is other than [0,1] or [1,0] -> throw an error

3.5 Reset:

- **reset() -> resets entire sheet**

    - ➢ If template sheet name does not exist -> create a new sheet template sheet name .

- If the data is written outside the specified range -> If the sheet exists the whole sheet gets cleared, else if the new sheet is not present then a new sheet will be created and data is stored in it.

4 Conclusion

In conclusion, the Student Elective Course allocation system represents a significant step forward in simplifying the complex task of managing elective course allocation for students. This solution uses Google Apps Script to create an efficient and streamlined process within Excel, empowering administrators to make well-informed and effective decisions with minimal technical knowledge.

With the successful implementation of this system, the burden on administrators is greatly reduced, enabling them to allocate elective courses to students with ease and minimizing the chances of errors that can occur with manual allocation.As we move forward, there is room for further improvements and customization, and the potential for addition of new features.