API documentation

Gsoc project idea

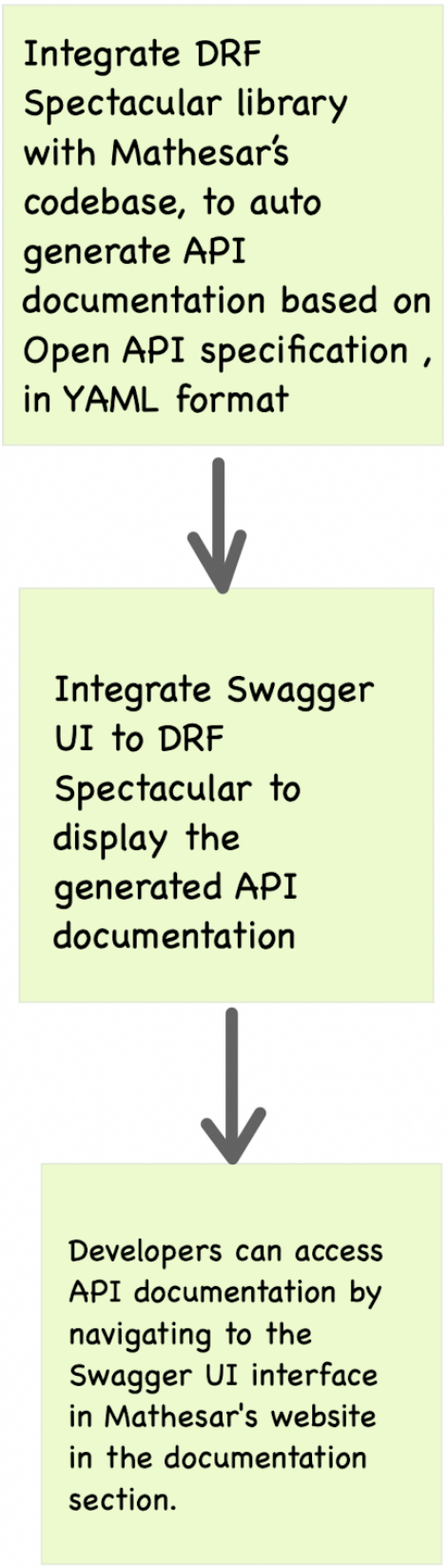
Synopsis

The aim of this project is to automate the generation of a well-written and easy to understand API documentation for Mathesar.It can help attract more developers to Mathesar's community. This can lead to more contributions, bug reports, and overall improvements to the software.It would also enhance collaboration between front-end and back-end developers.In addition to that, it would save the time of core members of Mathesar, as the API documentation can help new developers to understand API's functionality and structure.In this project, we will use DRF's Spectacular and Swagger UI for API documentation as it can generate and maintain API documentation automatically.This will help reduce the manual work required to do the same.

Implementation

Currently there is no proper API documentation and is done only manually.This project aims at automation of API documentation generation for Mathesar's current and upcoming APIs in a well-defined standard that can be easily understood by the developers.To achieve this, we will use a library, DRF Spectacular, which can generate Open API specification automatically based on the backend implementation. Open API is a well-defined standard supported by a large community and provides a machine-readable format for describing the functionality of the API. Additionally, Open API documentation can be easily consumed in different formats like JSON or YAML.We will generate it in YAML format,as it is a more human-readable format and would enable developers to use any api platform of their choice(like postman, hopscotch, etc).
Swagger UI is a tool that can be used to visualize and interact with the Open API documentation. It provides a user-friendly interface for exploring the API endpoints, parameters, and responses.We will integrate Swagger UI to DRF Spectacular, to display the API documentation generated by DRF Spectacular.

FLOW OF IMPLEMENTATION:



API spec

Mathesar uses JSON content-type API to communicate with the frontend client.There are several specifications available for documentation of Mathesar's RESTful APIs.The common ones used in industries are OpenAPI, RAML, API Blueprint and JSON API.Out of all these options, we choose
OpenAPI specification to document Mathesar's API.
OpenAPI Specification (formerly known as Swagger Specification) is an open-source format for describing and documenting APIs.OpenAPI allows developers to define the structure of their API, including endpoints, parameters, responses, and authentication methods, using a JSON or YAML file. This makes it easier for developers to understand how to use the API.

Why are we using OpenAPI Specification over the other formats available?

- OpenAPI provides a comprehensive set of features for documenting RESTful APIs
- It includes support for both JSON and YAML formats
- It is easy to integrate with Mathesar's Django and Django rest framework
- OpenAPI provides interactive documentation tools that allow developers to test API endpoints and see the responses in real-time.

But one of the main reasons we use OpenAPI Specification in our project is because:

- Django rest framework used in Mathesar provides built-in support for generating OpenAPI documentation and validating API requests and responses against an OpenAPI definition.
- There are good tools available namely, Swagger UI and ReDoc for rendering OpenAPI documentation in a web browser

Architectural / UX problems

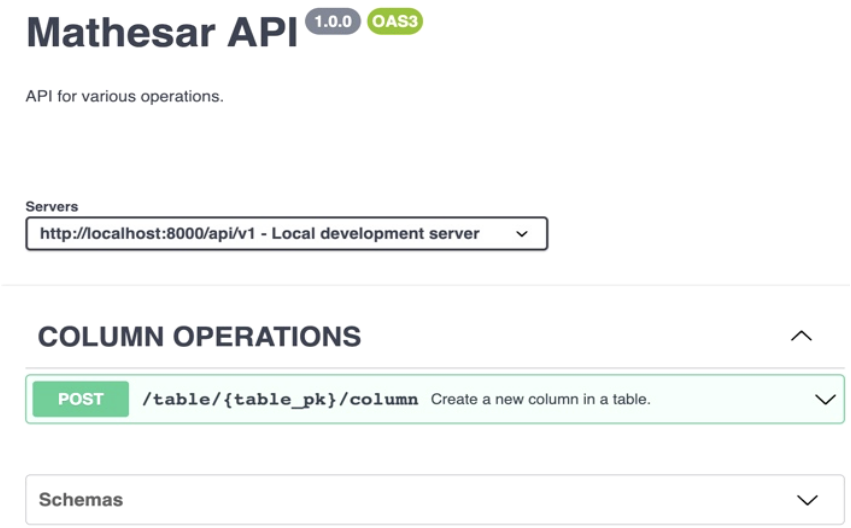
The main architectural problem currently existing in Mathesar is the improper API documentation,which makes it difficult for developers to understand and use the API.Since there was no automatic tool for generating API documentation in Mathesar, the only way to document the API was manually, which is time-consuming and error-prone.To address this problem,the current architecture of the Mathesar codebase will not be significantly changed. Instead, we will be adding a drf spectacular library to the codebase that will automatically generate API documentation based on the existing code. Drf spectacular will interact with the existing codebase through the Django Rest Framework API views, serializers, and models.This information will be used to generate OpenAPI documentation, which can then be served using Swagger UI

Given below is the sample API documentation specification generated by DRF Spectacular for **creating a column** in the table, in Mathesar, in YAML format..

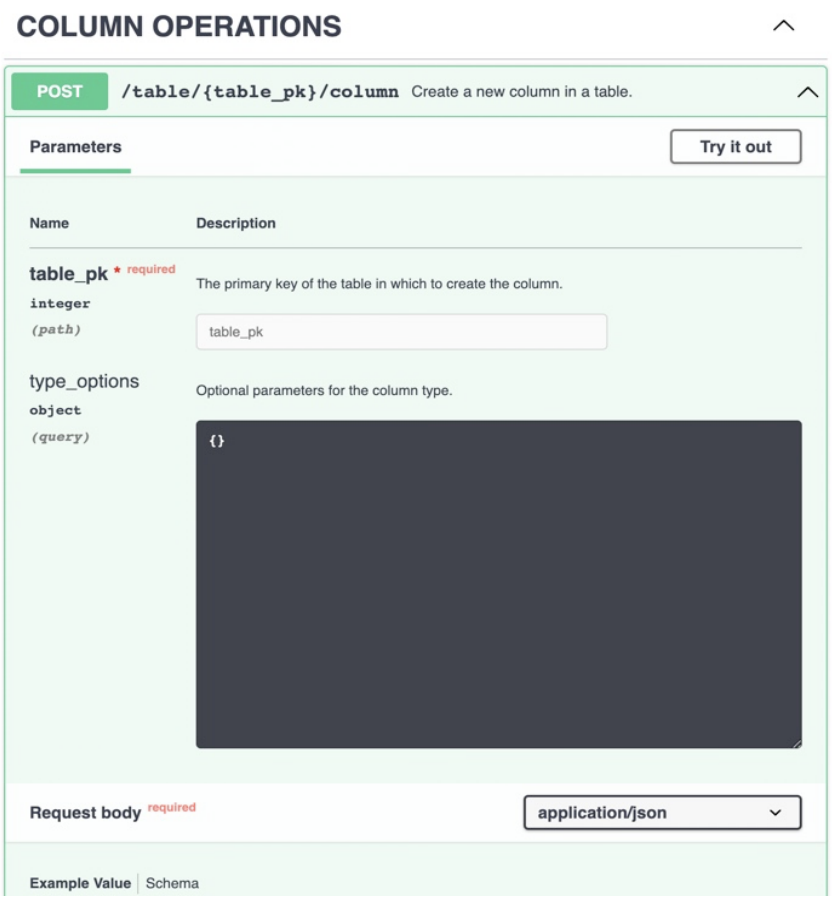
```yaml
openapi: 3.0.0
info:
  title: Mathesar API
  description: API for various operations.
  version: 1.0.0
servers:
  - url: http://localhost:8000/api/v1
    description: Local development server
  - url: https://api.example.com/v1
    description: Production server



paths:
  /table/{table_pk}/column:
    post:
      operationId: create_table_column
      summary: Create a new column in a table.
      tags:
        - COLUMN OPERATIONS
      parameters:
        - in: path
          name: table_pk
          schema:
            type: integer
          required: true
          description: The primary key of the table in which to create the
            column.
        - in: query
          name: type_options
          schema:
            type: object
          required: false
          description: Optional parameters for the column type.
      requestBody:
        required: true
        content:
          application/json:
            schema:
              $ref: '#/components/schemas/ColumnSerializer'
            description: The column to be created.
      responses:
```

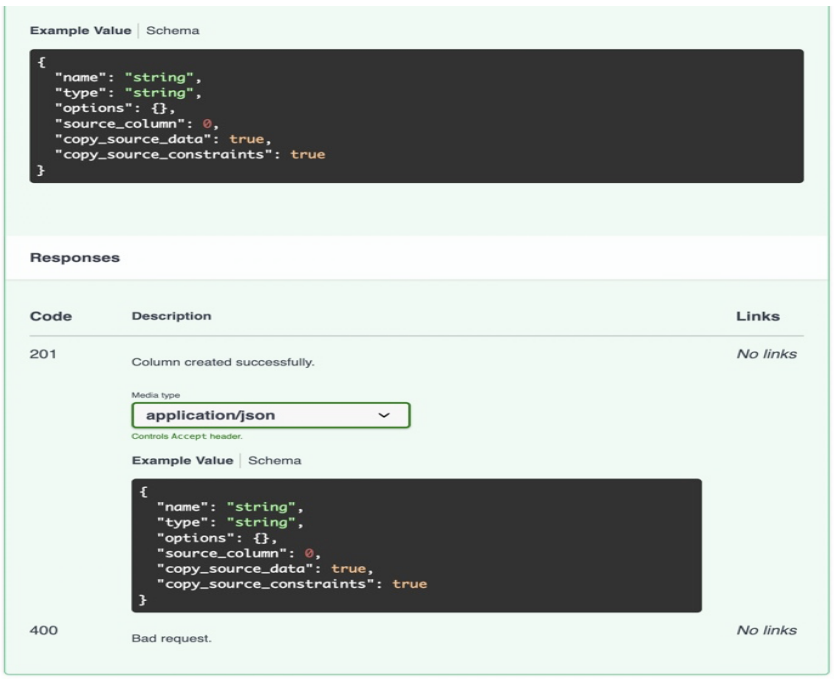
```yaml
      responses:
        '201':
          description: Column created successfully.
          content:
            application/json:
              schema:
                $ref: '#/components/schemas/ColumnSerializer'
        '400':
          description: Bad request.
components:
  schemas:
    Column:
      type: object
      properties:
        name:
          type: string
          description: The name of the column.
        type:
          type: string
          description: The data type of the column.
        options:
          type: object
          description: Additional options for the column type.
    ColumnSerializer:
      allOf:
        - $ref: '#/components/schemas/Column'
        - type: object
          properties:
            source_column:
              type: integer
              description: The primary key of the column to use as a source
                for copying data and constraints.
            copy_source_data:
              type: boolean
              description: Whether to copy data from the source column.
            copy_source_constraints:
              type: boolean
              description: Whether to copy constraints from the source column
                .
          required:
            - name
            - type
```

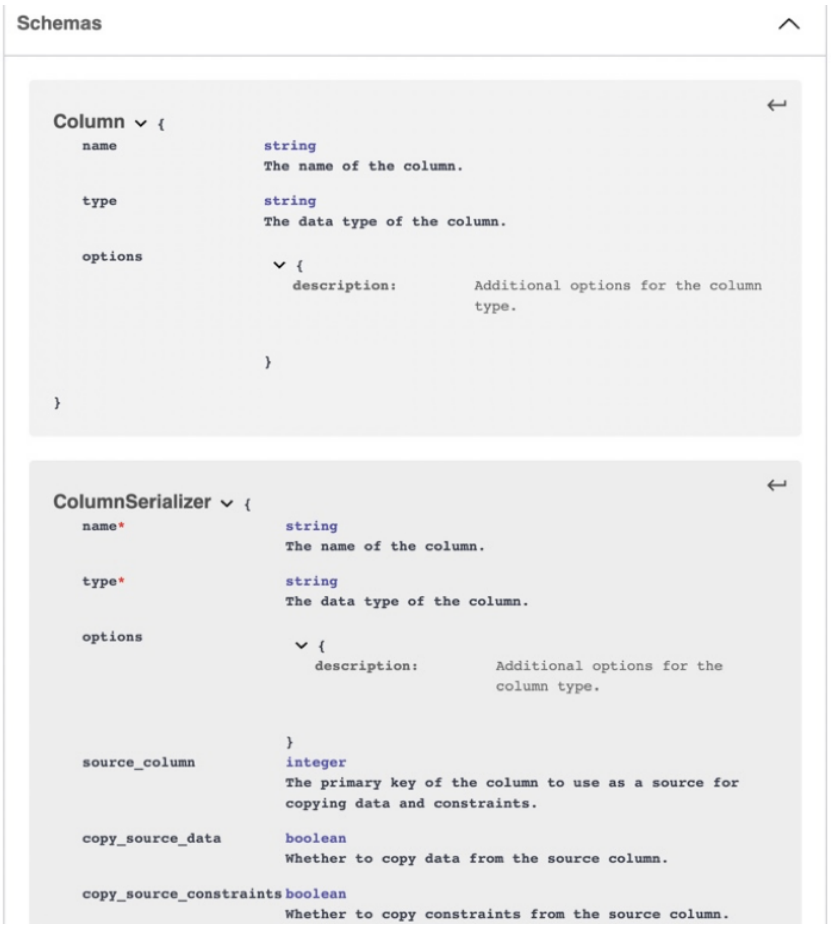
The API Documentation that would be rendered in Swagger UI would look like:



On clicking on the end point and expanding it ,additional information of the API will be displayed as follows:







What if the inferred APIs are incorrect ?

Taking the example of the same **Create column** API, as shown above, we have generated the API documentation but the generated schema is lacking Error schema.

```
Error ⌄ {
    message*            string
                        A description of the error.

    field               string
                        The field that caused the error.

    status_code*        integer
                        The HTTP status code of the error.


}
```

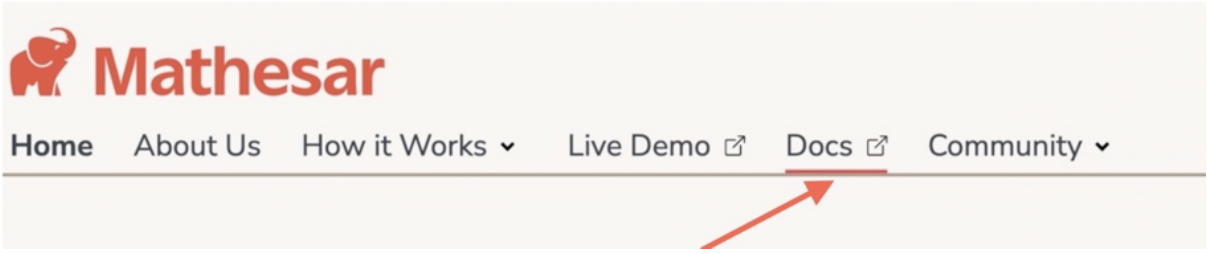
In that case, we will have to manually override the schema.We can do this in the following ways:

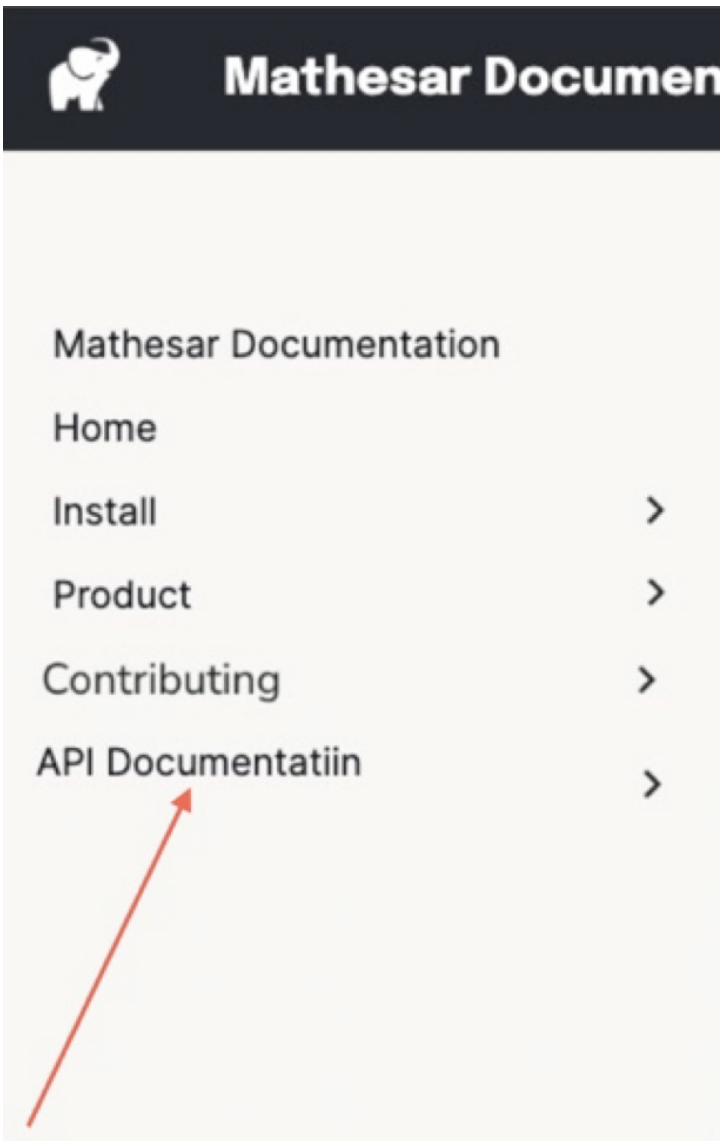
METHOD 1: DRF Spectacular provides several OpenAPI decorators, such as @extend_schema, @override_method, @parameter, and @response, which can be used to add, modify, or remove information from the generated OpenAPI schema.
If the customised schema are still not satisfactory, DRF Spectacular also allows to declare the schema manually using Python classes. We can create a custom schema by subclassing one of the provided schema classes, such as AutoSchema, and override its methods to customize the schema generation process.
Taking the **above example**, we will use @extend_schema to add the schema for **Error**.

METHOD 2: The second way is to manually update the OpenAPI specification file. The OpenAPI specification file is a YAML file that defines the API endpoints, parameters, responses, and other details. We can manually edit this file to add, modify, or remove the endpoints and their details. Then, when we regenerate the documentation, it should reflect the changes we made.

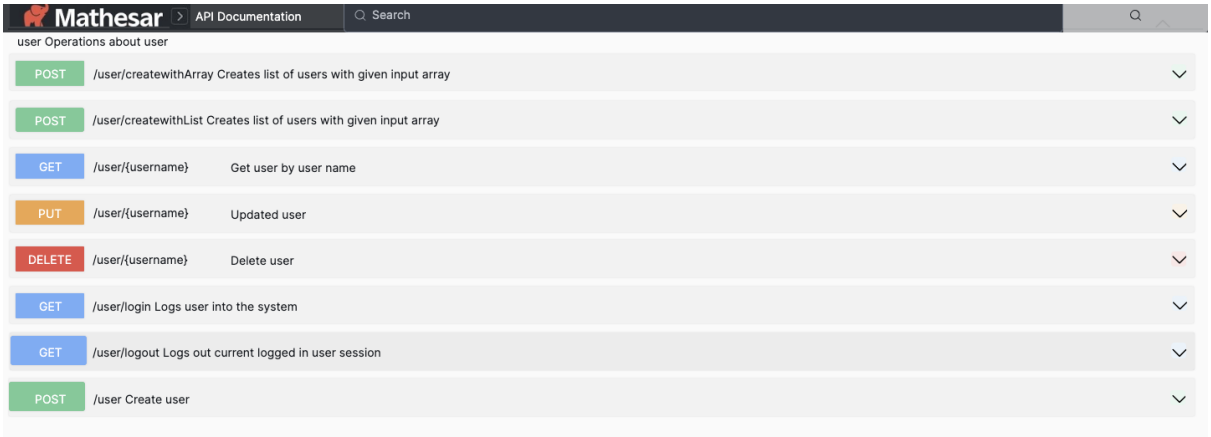
UX design

In order to render the API documentation generated by DRF Spectacular, we will use Swagger UI.Swagger UI is an open-source tool that allows developers to visualize and interact with the documentation of an API. It provides a user-friendly interface that enables developers to explore the available API endpoints, request parameters, responses, and schemas that has been generated by DRF Spectacular.Not just that, Swagger UI also allows developers to explore and test the API endpoints directly from the documentation.We developers can access the API Documentation by navigating to the docs section of Mathesar's website as shown below.





In the local navigation, “API Documentation” section will be added.On clicking this, the users will be redirected to Swagger UI. The Swagger UI will be embedded directly into the Mathesar UI. On the Swagger UI page, the user can explore the different endpoints and methods available in the API.They can also view example requests and responses, and try out API calls directly from the UI.They can also view example requests and responses, and try out API calls directly from the UI. The section can include a search bar at the top to help users find specific endpoints or methods. Once the user selects an endpoint or method, the relevant information will be displayed.
We will add custom branding to the Swagger UI page to make it feel more integrated with the Mathesar UI. This will include adding the Mathesar logo, customizing the color scheme, typography, and layout and adding some explanatory text about how to use the API.The mockup for the same is given below:



Overall, the goal of the UI changes would be to provide users with a seamless experience when navigating between Mathesar and Swagger UI.

External Dependencies

Dependency	Purpose	License	Reason for Selection
Django Rest Framework Spectacular	Generate OpenAPI specification for Mathesar API	BSD-3-Clause	This library is specifically designed to work with Django Rest Framework and provides a simple way to generate OpenAPI specifications for APIs that are easy to read and understand. It is well-documented and well-maintained,has a good community and the BSD-3-Clause license is compatible with Mathesar's GPLv3 license.
Swagger UI	Provide user-friendly interface for API documentation	Apache-2.0	Swagger UI is a popular and widely-used library for API documentation, and provides a visually appealing and allows end developers to effortlessly interact and try out every single operation our API exposes for easy consumption .The Apache-2.0 license is also compatible with Mathesar's GPLv3 license.
PyYAML	Library used by DRF Spectacular for parsing and generating YAML files	MIT	This format allows developers to use any API platform of their choice(postman, hopscotch, etc).The MIT license is compatible with Mathesar's GPLv3 license.

Research & References

- DRF Spectacular GitHub page: https://github.com/tfranzel/drf-spectacular
- Open API Specification website: https://www.openapis.org/
- DRF's "Documenting Your API" page
- swagger.io/docs/

Timeline & Deliverables

TIMELINE	DELIVERABLES
May 4 - 28 **Community bonding period**	Plan for getting up to speed on the Mathesar codebase and relevant technologies (Django, DRF, OpenAPI, DRF Spectacular, Git, Swagger ui).
Week 1-2: May 29-June12	• Integrate DRF Spectacular with Mathesar backend codebase. • Customize the API documentation output to YAML format. • Generate the schema file • Generate the automated API documentation
Week 3-4: June 12-26	• Set up Swagger UI and integrate with DRF Spectacular • Add custom decorators to allow manual override of the inferred APIs
Week 5-6: June 26-July10	• Create mock API endpoints for testing. Write tests to ensure the API documentation and Swagger UI are working as expected. • Prepare for mid term Evaluation
Week 7- 8: July 10-24	• Well-designed UI for API documentation page, this includes custom branding to the Swagger UI page. • Finalize the design and layout of the API documentation page. • Add descriptions, examples, and responses to the API documentation
Week 9: July 24-31	Improve documentation quality, perform code review and refactor, address feedback and feature requests
Week 10: July 31-Aug 7	Buffer week for any delays or unforeseen issues
Week 11: August 7-14	Finalized API documentation generator and code cleanup
Week 12 : August 14-21	• Address any remaining issues or bugs. • Documentation of integration, testing, and UI design

August 21-28	Submit the final work product and final mentor evaluation

Questionnaire

- Why are you interested in working on Mathesar?

I am an open source enthusiast.Mathesar has a growing community of contributors, which means that there are opportunities to learn from others and collaborate on new features and enhancements.
In addition to that,Mathesar has the potential to make a significant impact in the database management space by democratizing access to databases and making it easier for people to work with data. Contributing to Mathesar can be a way to make a positive impact on the world.
Also,Mathesar is built using modern web development technologies such as Django, Django Rest Framework. Working on this project provides an opportunity to gain experience with these technologies and build skills in modern web development practices.

- Why are you interested in working on this project idea?

Improving the API documentation of Mathesar can have a real-world impact by making it easier for developers to understand and develop the software, which can lead to increased productivity and efficiency.This project involves working with technologies such as Django, Django Rest Framework, and OpenAPI, which will be useful to develop my tech stack.

- What about your skills and experience makes you well-suited to take on this project?

I have a good understanding of the Django web framework and the Django Rest Framework as I have built a blogging website using the same.I have attached the GitHub repo link in the experience section.I am familiar with OpenAPI specification for building APIs, and the ability to work with the OpenAPI format.I am also comfortable integrating and customising Swagger UI.I have the ability to pay close attention to details,as small errors or omissions in API documentation can cause significant problems for developers.In addition to that,I have good communication skills for effective collaboration and contribution to Mathesar.

- Do you have any other commitments during the program period? Provide dates, such as holidays, when you will not be available.

I do not have any personal commitments or obligations, and I am available to work on this project throughout the Google Summer of Code program period. I do not have any scheduled holidays or planned absences that would affect my availability.I will commit my full time in this project and can spend at least 30 hours per week.I am willing to dedicate more time if needed.

- If your native language is not English, are you comfortable working closely with a mentor in English?

Yes, I am very comfortable working closely with a mentor in English.

- Have you worked on a project remotely and/or with people in other timezones before? If you have, please provide details.

I do not have personal work experiences in working with individuals from different time zones.However,I can adapt to different working hours.

- Are you interested in contributing to Mathesar after the program is complete?

Yes! I have a strong interest in contributing to Mathesar after the completion of Google Summer of Code.I will make sure to contribute regularly and if there are any ongoing tasks or projects , after GSoC, that you think I would be a good fit for , please let me know.

General Information

About Me

I'm Varsha D R , a second year computer science student at M S Ramaiah University of Applied Sciences with experience in web development and Django. My major programming languages are Python and Java. I am also familiar with Django Rest Framework and have experience creating REST APIs.I have always manifested a strong academic record and nurtured with zeal for earning, I've never failed to miss the opportunity to learn more.As someone who is passionate about open source, I am committed to the principles of collaboration, transparency, and community-driven innovation.

Contact Information

- Full name: Varsha D R
- Email address: varshadr1234@gmail.com
- GitHub username: github/varshadr
- Personal website (if you have one): -
- Phone number: +91 9008216768
- Emergency contact information: +91 9740453001

Education

- Institute: M S Ramaiah University of Applied Sciences
- Degree: Btech
- Major: Computer Science
- Graduation year: 2025
- Courses taken: Information Science

Skills

Skill name	Proficiency (1-5)	Where you've used this skill
Python	*4*	*As part of college coursework and side project*
Django	*4*	*As a part of side project*
Django REST framework	*3*	*As a part of side project*
Git	*5*	Contributions to Open source projects.
Github	*5*	Hosted my side project repositories and contribution to open source
Gitlab	*5*	Contributing to Open source projects.

HTML	*5*	*As a part of side project and for LinkedIn skill assessment*
CSS	*3*	*As a part of side project*
JAVA	*4*	*As part of college coursework and by learning data structures and algorithm*
PHP	*3*	As a part of side project
CakePHP	*3*	As a part of side project.

Experience

Brief description	Relevant links	Additional notes
PR merged to open source project at CDLI	*https://gitlab.com/cdli/framework/-/issues/1480*	Added links to resources under resouces
(I will fill this section before submitting the final proposal)		

Contributions to Mathesar

I am still trying to actively contribute to Mathesar.

Issue title	Links to issue and/or PR	Additional notes
Solved issue #23: "Example issue name"	*Link to issue, Link to PR*	-