Automating member interaction

For Mentoring Europe

Maksim Kornakov

Company Introduction

Mentoring Europe

- Non-profit organisation that fosters the growth and effectiveness of mentoring practices throughout Europe.

- Membership program exists and connects mentoring practitioners to share findings and knowledge and participate in events and conferences.

- The company also helps small mentoring programs to apply for grants such as Erasmus+ by complying to all the criteria and formalities.

Focus points

- Invoices to members are handled manually
- Membership certificates are created and sent manually
- Workforce rotation - only a few employees and many rotating interns
- Archive data is weakly structured (membership tables, etc.)
- Members delay payments

Approach

- Use my programing skills, extensive personal experience in the company as an intern, communication with employees, members and interns and secondary research to focus on several weak points:
    - Manual system that can be automated
    - Structure the archive data and make it error proof
- Identify the criteria for required changes and create a work plan supported by primary and secondary research
- Test the prototype, present it to the company, receive feedback and possibly implement necessary adjustments

Theoretical context

1. Identify well suitable theories for the project

- SWOT - Initial analysis and further impacts
- TOM - Technology acceptance model, to align the product with the organization

2. Research existing application, theory extensions and explanations

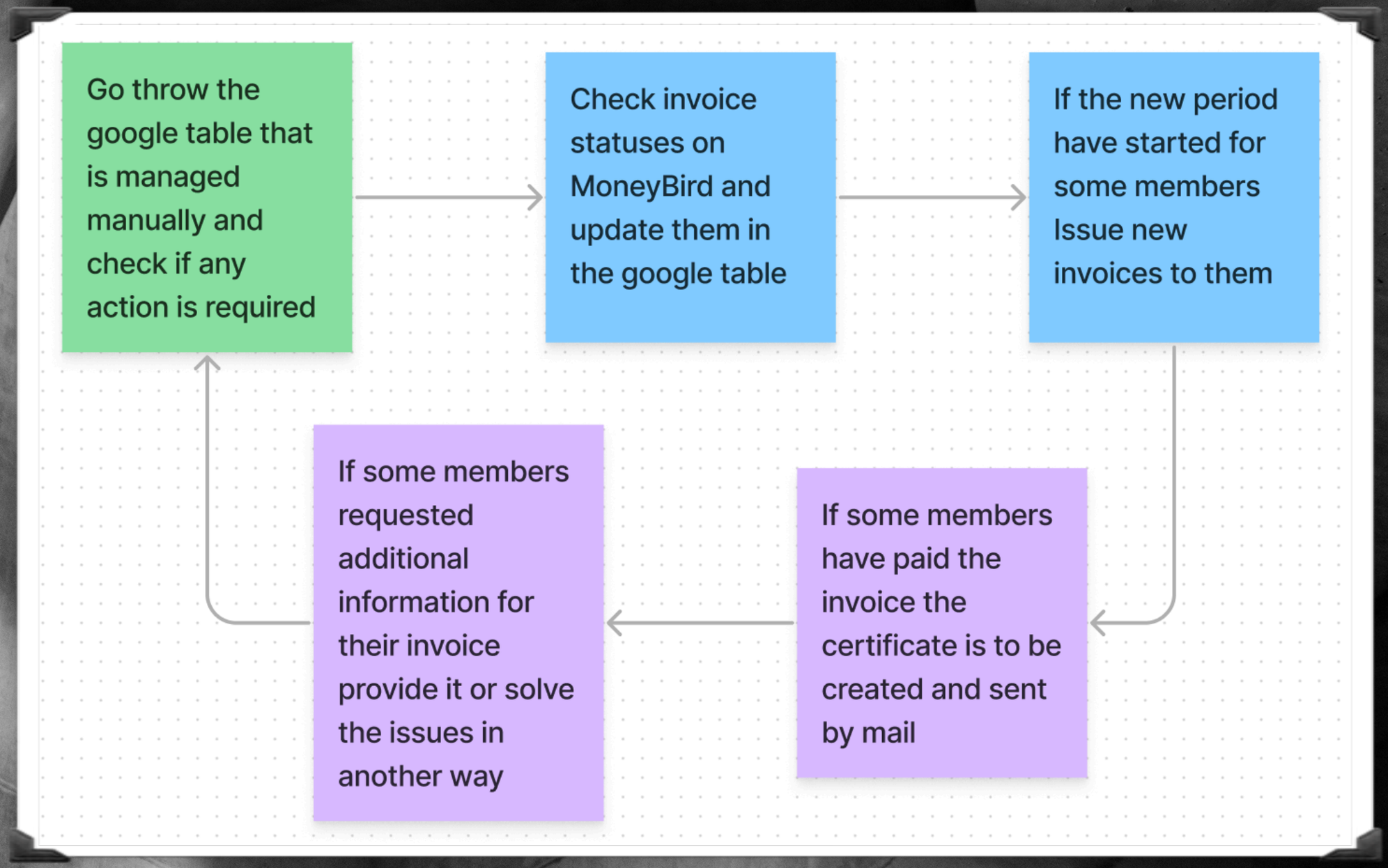
- Identify intersections with the theory, address weak points
- Reflect referring to the theory and applying it over the finished product

Technical Context

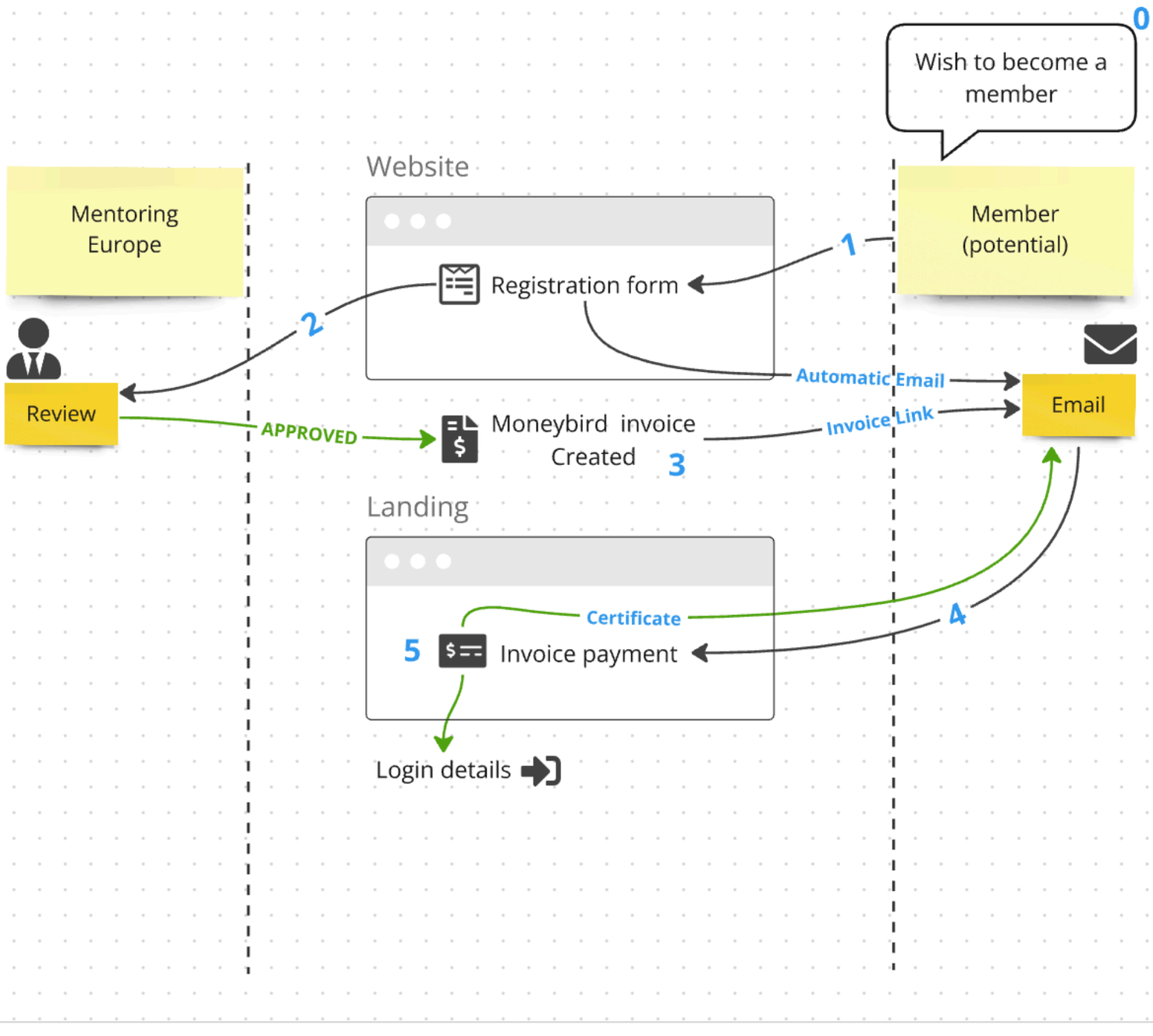
The product for Mentoring Europe is a program written in Typescript and is designed to run in a local/server Node.js environment. Therefore the best was done to describe how it works and what stand behind it for people without deep background in the specific topic.

Typescript - programing language for web
Node.js - environment to run servers

Current architecture



Proposed architecture



Description

Most of the manual repetitive tasks are handled automatically by the system.

Critical parts such as: Details input and overview control is left outside a closed system to be verified by the manager or dedicated employees.

Explanation

The main system lives on a server on the internet, because of that it is able to communicate with various systems automatically. For that API is used, basically a set of instructions in common format that are sent to Google, MoneyBird or any other system to retrieve data or perform an action.

The system is responsible for retrieving needed data on time, deciding on whether to perform any action and do so. Besides that 2 Google spreadsheets were created:

1. Control table, to track members, and payments
2. Full overview table with all member details, payments, delays, etc.

The first one is only accessed by the manager, the second one is readonly and can be accessed by employees and interns to retrieve specific details.

Control and overview tables

Description

Manger inputs member detail into the control table with minimum needed details to start automatically processing members.

The overview table is redesigned using and existing overview to keep well-known structure and workflow. However it is made readonly to eliminate typos and errors.

Active Members

ID	Contact Person	Start Date	Status
4	John Johns	01.2024	invoice sent
3	Other At me	02.2024	invoice sent
2	Maksim Kornakov	03.2024	invoice sent
			none
			invoice sent
			paid
			certificate sent

	A	B	C	D	E	F	G	H	I
1	Active Members Overview								open | paid | none
2	**Member ID**	**Name**	**Invoice Email**	**Contact Person**	**Address**	**Postal Code**	**City & Country**	**Start date**	**State**
3	**4**	Spanish test member	jhon@mkor.dev	John Johns	Some street	8923AA	Leeuwarden, NL	01.2024	invoice sent
4	**3**	ME Member	member@mkor.dev	Other At me		8917JA	Leeuwarden, NL	02.2024	invoice sent
5	**2**	test for api	testInvoice@mkor.dev	Maksim Kornakov			, NL	03.2024	invoice sent
6									
7									

Benefits

1 Cost Savings

Frontend - Free
Backend - 6$ / month

2 Simplified Content Management

Content updates and additions will be more straightforward, reducing the time and effort needed

3 Consistent Design

The website will have a cohesive and professional design throughout, improvin
user trust and engagement

4 Enhanced Security

The modern setup will provide better protection against online threats, safeguarding user data and website integrity

5 Efficient Process Automation

With the proposed setup, processes such as sending emails and certificates can be automated, saving time and ensuring seamless communication and recognition for members.

Feedback from the company

Positive

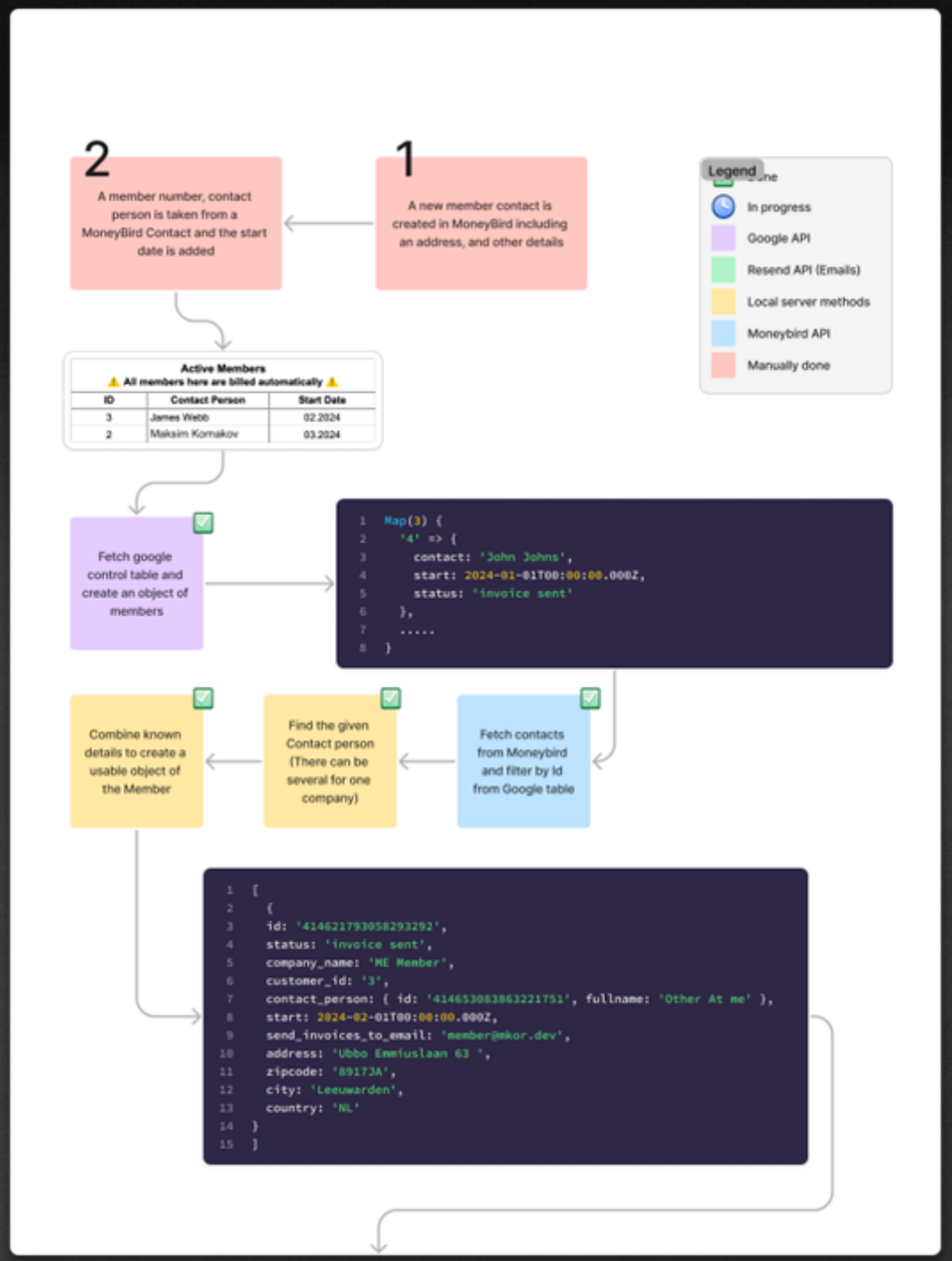
- Clear architecture and task separation
- Increase competency level among company interns as they can focus on more interesting and developing tasks

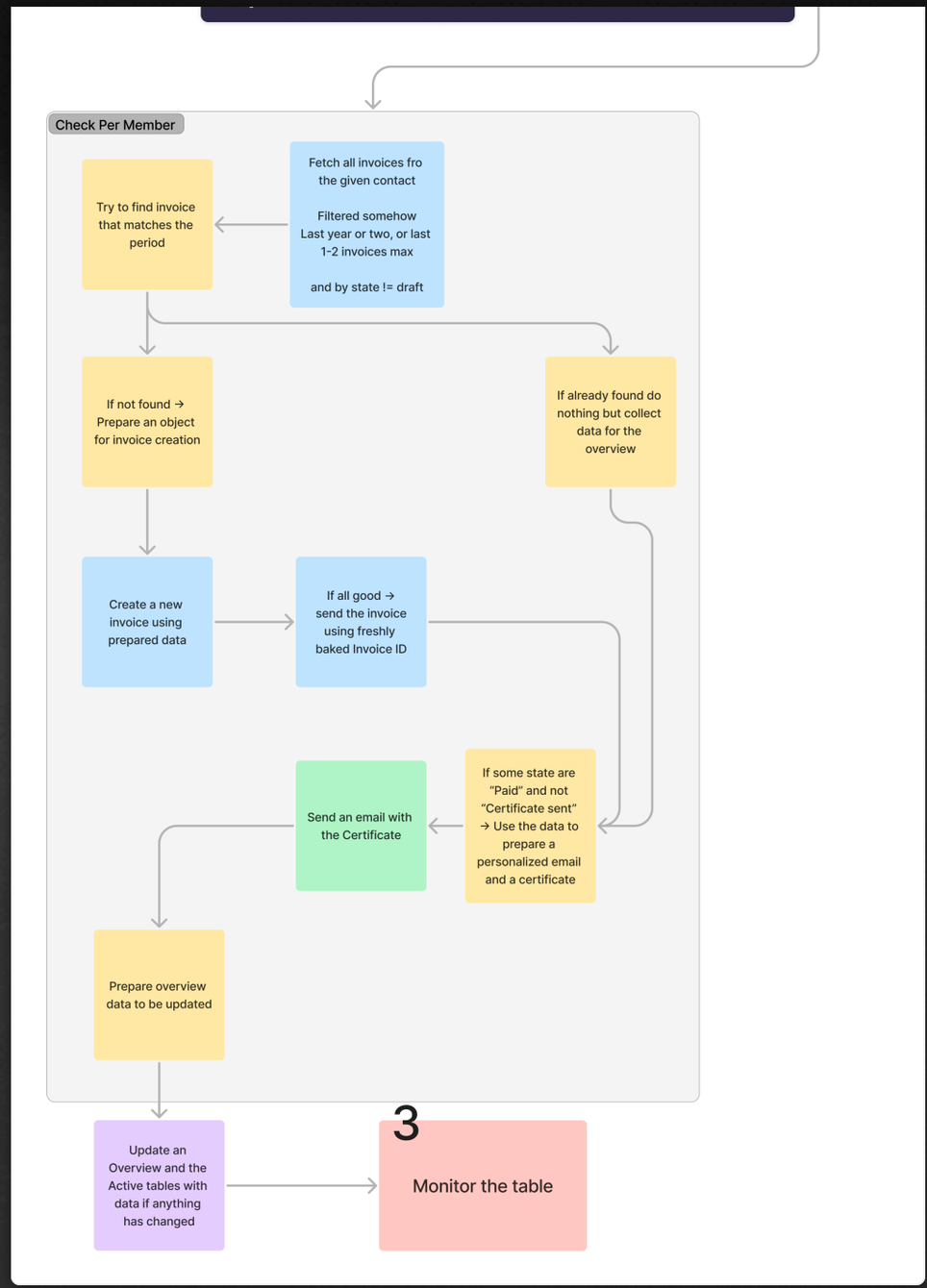
Negative

- Complicated technical system
- Current data is chaotic and not suitable to input into such a system

Technical details

Prototype architecture, Tested in sandbox environment





Conclusion

Comparison table

Aspect	Before	After
Invoice generation	Manually created per member by intern, employee or manager.	Automatically handled via the system, triggered according to the control table.
Overview table	Manually edited, prone to errors.	Read-only; automatically updated and synchronised with MoneyBird.
Responsibility for Invoicing	Intern/employee.	The manager initiates the periodic invoicing, and interns and employees have access to a global overview table.
Security and access control	A responsible person requires access to the company's MoneyBird financial account.	Only the manager handles sensitive data; interns interact with simplified overviews.
Error risk & Consistency	High, dependent on manual accuracy and timing.	Low, automation ensures data is synced and accurate.
Technical management	Ongoing manual handling, some logic described in the onboarding manual.	Systemised logic; should require minimal tech skills to configure most of the settings.
Time per Invoice	5+ minutes manually.	Essentially no time consistently (once triggered, the process is automated)