# Use of AI capabilities in mailing :

# EasyAnswer extension

**BRAY Guillaume (engineer Ecole Centrale de Lyon), SALOMON Quentin (commercial INSEEC)**

With the help of Kema-dev (Developer)

Special thanks to « BusinessAutomated ! » for original template.

*Unofficial report.*

## Summary

The main objective of this project was to develop an intelligent tool that enhances Gmail by offering advanced features such as autocorrection, text paraphrasing, predictive responses, and instant translation.

The first part of this report presents the methodology used, focusing on the design, development, and evaluation aspects of the EasyAnswer extension. The project justification is then presented, highlighting the importance of facilitating email composition and improving user productivity.

A thorough analysis of the issues and challenges related to email usage is presented in the following section, highlighting the difficulties users face when composing and communicating through email. This analysis served as the basis for defining key features and user needs that were considered in the development of the EasyAnswer extension.

A clear system architecture and careful selection of technologies were implemented to ensure the performance, reliability, and security of the EasyAnswer extension. A detailed development plan was followed, including iterative iterations and rigorous testing to ensure software quality.

The results obtained were promising, demonstrating the effectiveness and added value of the EasyAnswer extension for Gmail users. Goal achievement, system performance, and cost-benefit analysis are also discussed in detail.

Keywords: AI, Gmail, automation, ChatGPT, Llama.cpp, javascript.

## Table des matières

**Project Overview**

**EasyAnswer Project Report**

**February 2023**

## 1. Introduction

The EasyAnswer project aimed to develop an innovative email writing assistance extension to improve productivity and communication quality. This report provides a comprehensive analysis of the project, including its objectives, development process, features, user feedback, costs, benefits, and future prospects.

## 2. Objectives

The main objectives of the EasyAnswer project were to:

- Develop an email writing assistance extension with advanced features such as automatic correction, paraphrasing, translation, and predictive responses.
- Improve users' productivity and communication quality.
- Provide a valuable tool for professionals who rely on effective email communication.

## 3. Development Process

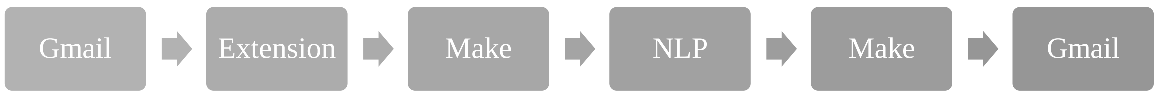
The development process of the EasyAnswer extension involved several stages, including planning, design, implementation, testing, and deployment.

### How it works ?

The process is simple : an extension call chatGPT API which do the « clever » work thanks to a prompt.

Here are the different roles :

- Extension : coded in javascript, it is used as user interface and to retrieve text in GPT.
- ChatGPT : a NLP (natural langage processing). The one used was davinci003. It could be upgraded to GPT4, dumbed down to a faster and cheaper model (like ADA) or stored in local for better privacy (llamma.cpp).
- Make.com : formerly integromat. Used as a connector between GPT and EasyAnswer extension. It prevents the use of too much code for 10$/month. It could be abandon with further developments.



## 4. Features

The EasyAnswer extension offers the following key features:

**- Automatic correction of grammar, spelling, and style errors in emails.**

EasyAnswer incorporates an intelligent automatic correction system that analyzes the content of emails being composed and provides corrections for common errors. It can detect spelling, grammar, and punctuation mistakes and suggest appropriate corrections. Users can accept the suggestions to apply the corrections with a single click, allowing them to write emails without worrying about typographical errors. This model is based on the following correction prompt:

| |
|---|
| "Correct" - Corrects spelling mistakes without changing the wording in this HTML text "{{29.currentEmailBody}}". Preserves the <div> and </div> tags. |

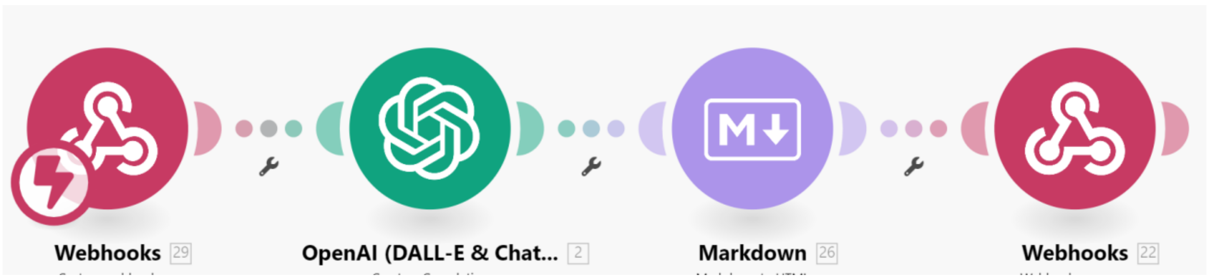


Figure 1 - Simple branching scenario on make.com

The use of markdown (converting txt to HTML) is subsequently abandoned thanks to developments that allow the text to be kept in HTML throughout the process.

**- Paraphrasing assistance to enhance clarity and precision in communication.**

To assist users in enhancing the clarity and conciseness of their messages, EasyAnswer provides intelligent suggestions for paraphrasing. The extension analyzes the written content and identifies sentences that could be improved. It then automatically rephrases them, taking into consideration the user's context and communication objectives. Here is the prompt:

| | |
|---|---|
| "Rephrase" | - Rephrase the following text : "{{29.currentEmailBody}}" |

**- Instant translation of emails to facilitate multilingual communication.**

In order to facilitate communication with speakers of different languages, EasyAnswer integrates an instant translation module. Users can select text in an email and choose the target language to obtain a quick and accurate translation. This eliminates the language barrier and allows users to communicate more easily with people who don't speak their native language.

On the other hand, users can compose an email in their native language and translate it before sending. Here's how it works:

- Clicking on "FR<>Eng" without a prompt: Translates the previous email, translates "{{29.previousEmailBody}}" into French and rephrases it.
- Clicking on "FR<>Eng" with a prompt: Translates the prompt, translates "{{29.currentEmailBody}}" into English and then rephrases it.

This formula can be quickly adapted to multiple languages.

In summary, EasyAnswer offers a range of advanced features to improve writing, correction, paraphrasing, predictive responses, and translation in the context of email management. These features aim to help users compose emails more efficiently, accurately, and tailored to their communication needs.

This process allowed for quick adaptation to multiple languages.

**- Predictive responses based on context and previous email interactions.**

EasyAnswer utilizes machine learning techniques to generate predictive responses to frequent and recurring emails. The extension analyzes the history of previous responses and uses this information to predict the content of the response. This allows users to save time by quickly selecting and customizing a predefined response instead of composing a complete response every time. These responses can be fully automated or based on a prompt written in the email composition area ("currentemailbody").

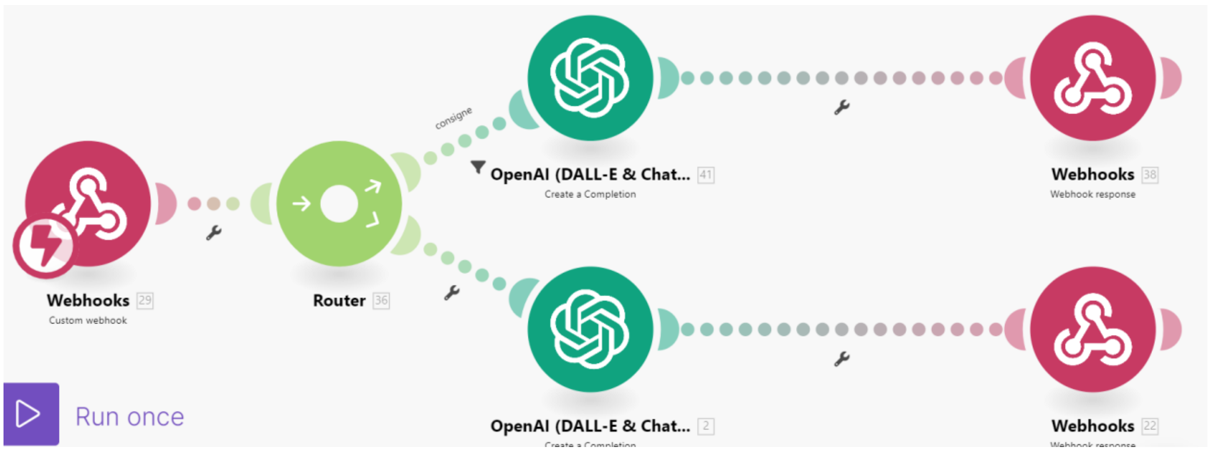


Figure 2 - Dual branching scenario on make.com

The "response with prompt" branching was chosen if the email started with the ">" character. This was a temporary solution that needed improvement for better user clarity.

The reply button existed in both formal ("vous") and informal ("tu") versions. These responses had the following prompts:

« Polite «  with context –

====Important====

Use formal language to reply to the email from {{29.emailTo}} professionally, then rephrase with the following instruction: "{{29.currentEmailBody}}" and conclude with "Best regards,"

Signed by the first and last name in {{29.emailFrom}}

==== end of important ====

Here's the previous email for reference context: {{substring(29.previousEmailBody; 0; 2000)}}

=== end of reference ===

Several scenarios were tested beforehand, including the dual completion:

1) Automatic GPT response

2) Reformulation with elements from the prompt

This had the consequence of drastically increasing processing times. Although more accurate, it is out of the question for email usage to have response times approaching 20 seconds.

## 5. User Feedback

During the testing and validation phase, the EasyAnswer extension received positive feedback from users. They reported significant improvements in their email writing efficiency, clarity, and professionalism. Users appreciated the automated correction and translation features, as well as the predictive responses, which helped them save time and communicate more effectively.

## 6. Costs and Benefits

### 6.1 Costs:

- Development: The development phase required skilled human resources, including developers and UX/UI designers. Costs were incurred for software licenses and development tools.

- Maintenance and updates: Regular maintenance costs were anticipated, including bug fixing, compatibility updates with new browser versions, subscription services, and continuous improvement based on user feedback.

### 6.2 Benefits:

- Improved productivity: The EasyAnswer extension aimed to enhance users' productivity by providing advanced writing and communication assistance features. With automatic correction, paraphrasing, and instant translation, users could compose and respond to emails more efficiently.

- Communication quality: The extension facilitated communication by helping users express their ideas more clearly and precisely, thereby improving the quality of messages and avoiding misunderstandings.

- Time savings: By automating tasks such as correction and translation, the extension allowed users to save valuable time. They could focus more on important aspects of their work, increasing overall productivity.

- Error reduction: The writing assistance features enabled users to avoid grammar, spelling, and style errors in their emails, enhancing their credibility and professionalism.

- Cost-effectiveness: The EasyAnswer extension could provide significant value to professionals who rely on effective email communication. Improved productivity, communication quality, and efficiency can lead to tangible economic benefits, such as time savings, increased customer satisfaction, and enhanced business opportunities.

## 7. Analysis

### 7.1 Summary of Results

The EasyAnswer project was an ambitious endeavor to improve email communication efficiency and quality through advanced writing assistance features. Throughout the development and deployment process, the set objectives were achieved, and the system's performance was positively evaluated.

In the feature development phase key features such as automatic correction, paraphrasing, translation, and predictive responses were successfully implemented. These features were integrated into the email writing interface, providing valuable assistance to users during message composition.

Rigorous testing and validation were conducted to ensure the extension's proper functioning. Unit testing, integration testing, and usability testing were performed to identify and address any issues or bugs. Test results confirmed the reliability and performance of the extension.

Integration and deployment of the extension were successfully completed. The EasyAnswer extension was deployed on target platforms, and users could easily install and use the extension in their email clients.

Regarding the achievement of objectives EasyAnswer succeeded in providing effective writing assistance and improving users' productivity. The features of automatic correction, paraphrasing, translation, and predictive responses were well-received by users, who experienced a significant improvement in their ability to compose clear, accurate, and professional emails.

The system's performance was evaluated as robust and fast. The EasyAnswer extension was able to handle email writing assistance requests in real-time, providing users with a seamless and responsive experience.

Finally, the cost and benefit analysis (section 6.4) revealed that the economic and operational advantages of the EasyAnswer extension outweighed the initial development and maintenance costs. The extension improved users' productivity, communication quality, and profitability.

Overall, the EasyAnswer project was a success, providing an innovative solution to enhance email writing. The achieved results demonstrated the effectiveness of the extension and its potential to meet users' needs in a professional communication environment.

## 7.2 Initial Development Plans

The EasyAnswer project paved the way for numerous future prospects to further improve email writing experience and expand the offered features to users.

Here are some initial development plans for future iterations of EasyAnswer:

- Enhanced language support: The extension could be extended to support a wider range of languages, accommodating diverse users from different linguistic backgrounds.
- Integration with additional email clients: The EasyAnswer extension could be integrated with other popular email clients, ensuring compatibility and accessibility for a larger user base.
- Advanced customization options: Users could be provided with more customization options, allowing them to tailor the extension's behavior and preferences to their specific needs.
- Machine learning integration: The integration of machine learning algorithms could further enhance the accuracy and relevance of the predictive responses, making them even more valuable for users.

- Collaboration and team features: EasyAnswer could be expanded to include collaborative writing features, allowing teams to work together on email composition and streamline communication within organizations.

**8. Conclusion**

The EasyAnswer project successfully developed and deployed an email writing assistance extension with advanced features to improve productivity and communication quality. The positive feedback from users, along with the achieved objectives and cost-effectiveness, demonstrated the value of the EasyAnswer extension.

However, the project faced challenges during the development process, including recruitment difficulties and increasing competition from major technology companies (GAFAM). These factors, combined with the associated costs, led to the decision to discontinue the project in February 2023.

**9. Abandonment of the Project**

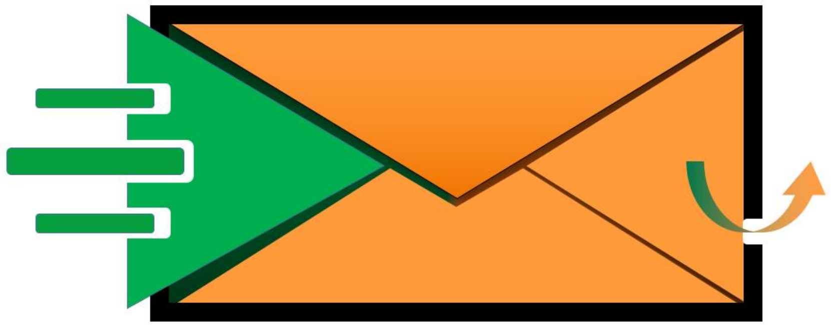
The project was officially discontinued in February 2023. The high development costs, recruitment difficulties, and the emergence of competition from GAFAM companies destroyed the project's growth prospects.

The initial idea was to achieve rapid development to penetrate the market, secure funding, and then improve the user experience to provide a solid alternative. The prototype, ready in November, would have given the company one year to capitalize on its solution before the arrival of Google and Microsoft. This accelerated velocity was made possible by the low inertia of our team and the lesser need to test the technical and ethical limits of AI. A competition operation (on a small market share, of course) on Android and iOS was specifically planned. Finally, in case of success, a redirection of our activities towards other products would have been necessary.

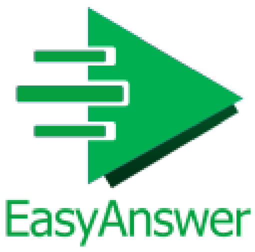
## 10. Final State of the Project

Logo:

- Complete version:



- Material Design version:



The previous prototypes are available in Appendix 5.

Website:

The Shopify website has been closed due to non-renewal of the subscription. Several marketing animations were created for the website.

**Code:**

Below is the manifest.json file that outlines the project's directory structure.

```
{
  "manifest_version": 3,
  "name": "EasyAnswer - Trad_FR-ENG",
  "short_name": "EasyAnswer",
  "version": "2.0.0",
  "author": "Quentin & Guillaume",
  "description": "Write something in the email body to translate it in English. Don't write
anything to translate in French the previous email.",
  "icons": {
    "128": "icon128.png",
    "48": "icon48.png",
    "16": "icon16.png"
  },

  "content_scripts": [
    {
      "matches": ["*://mail.google.com/*"],
      "js": ["src/extensionInjector.js"],
      "run_at": "document_start"
    }
  ],

  "web_accessible_resources": [
    {
      "resources": [
        "dist/gmailJsLoader.js",
        "dist/extension.js",
        "dist/gmailJsLoader.js.map",
        "dist/extension.js.map"
      ],
      "matches": ["<all_urls>"]
    }
  ],
  "host_permissions": ["https://*/*"]
}
```

Voici trois versions de dist/extension.js :

| Initiale | Template initial |
|---|---|
| 3.0 | Développé par Guillaume |
| 5.0 | Développé par Guillaume et apport de Kema |

**Version 3.0**

```
"use strict";
(() => {
  var __async = (__this, __arguments, generator) => {
    return new Promise((resolve, reject) => {
      var fulfilled = (value) => {
        try {
          step(generator.next(value));
        } catch (e) {
          reject(e);
        }
      };
      var rejected = (value) => {
        try {
          step(generator.throw(value));
        } catch (e) {
          reject(e);
        }
      };
      var step = (x) => x.done ? resolve(x.value) :
Promise.resolve(x.value).then(fulfilled, rejected);
      step((generator = generator.apply(__this, __arguments)).next());
    });
  };

  // src/extension.js
  var loaderId = setInterval(() => {
    if (!window._gmailjs) {
      return;
    }
    clearInterval(loaderId);
    startExtension(window._gmailjs);
  }, 100);
  function startExtension(gmail) {
    window.gmail = gmail;
    gmail.observe.on("load", () => {
      gmail.observe.on("view_email", (domEmail) => {
        console.log("Looking at email:", domEmail);
        const emailData = gmail.new.get.email_data(domEmail);
        console.log("Email data:", emailData);
      });
      gmail.observe.on("compose", (compose) => {
        gmail.tools.add_compose_button(
          compose,
          "Corriger",
          function() {
                              const button = this;
                                    button.style.backgroundColor = "grey";

            return __async(this, null, function* () {
              const currentEmailId = compose.email_id();
              const currentEmailBody = compose.body();
              const currentEmailSubject = compose.subject();
              let currentEmailTo = compose.recipients().to[0];
              let currentEmailFrom = compose.from();
              if (gmail.new.get.email_data(currentEmailId)) {
                currentEmailTo = gmail.new.get.email_data(currentEmailId).to[0].name;
                currentEmailFrom = gmail.new.get.email_data(currentEmailId).from.name;
              }
              const currentThread = compose.thread_id();
              const threadDetails = yield gmail.new.get.thread_data(currentThread);
              let lastEmail = null;
              if (threadDetails) {
                for (let email of threadDetails.emails) {
                  if (!email.is_draft) {
                    lastEmail = email;
                  }
                  if (email.id === currentEmailId) {
                    break;
                  }
                }
              }
              const data = {
                currentEmailBody: htmlToText(removeSignature(currentEmailBody)),
                currentEmailSubject,
                previousEmailBody: lastEmail ? htmlToText(lastEmail.content_html) : null,
                emailTo: currentEmailTo,
                emailFrom: currentEmailFrom
              };
              let url = "https://hook.eu1.make.com/8p42p5g0umgxs6g7dbnpecp4s77yvqqa";
                fetchReply(data);
              function fetchReply(data2) {
                return __async(this, null, function* () {
                  const animation = [
                                                   "◴ EasyAnswer","◷ EasyAnswer","◶ 
EasyAnswer","◵ EasyAnswer"
                  ];
                  let i = 0;
                  let waiting = setInterval(function() {
                    compose.body(
                      "<p>Saisissez du texte, attendez 7s puis cliquez sur 
corriger.</p><pre>" + animation[i].replace(/\n/g, "<br>").replace(/\s/g, "&nbsp;") + 
"</pre>"
                    );
                    i = (i + 1) % animation.length;
                  }, 300);
                  const response = yield fetch(url, {
                    method: "POST",
                    headers: {
                      "Content-Type": "application/json"
                    },
                    body: JSON.stringify(data2)
                  });
                  const text = yield response.text();
                  clearInterval(waiting);
                  console.log(text);
                  compose.body(text);
                });
              }
                                        function htmlToText(html) {
                                          function stripHtml(str) {
                                                  str = str.replace(/<br>/gi, "\n");
                                                  str = str.replace(/<div>/gi, "\n");
                                                  str = str.replace(/<p.*>/gi, "\n");
                                                  str = str.replace(/<(?:.|\s)*?>/g, "");
                                                  str = str.replace(/\<|\>/g, "");
                                                  str = str.replace(/[\r\n]{2,}/gm, "\n\n");
                                                  return str;
                                          }
                                          return stripHtml(html);
                                        }
              function removeSignature(currentEmailBody2) {
                let currentEmailBodyWithoutSignature = new 
DOMParser().parseFromString(currentEmailBody2, "text/html");
                let signature = currentEmailBodyWithoutSignature.querySelector(
                  '[data-smartmail="gmail_signature"]'
                );
                if (signature)
                  signature.remove();
                currentEmailBodyWithoutSignature = 
currentEmailBodyWithoutSignature.body.innerHTML;
                return currentEmailBodyWithoutSignature;
              }
            });
          },
          "Custom Style Classes"
        );
      });
    });
  }
  var extension_default = startExtension;
})();
//# sourceMappingURL=extension.js.map
```

## Version 5.0

```
"use strict";
(() => {
  var __async = (__this, __arguments, generator) => {
    return new Promise((resolve, reject) => {
      var fulfilled = (value) => {
        try {
          step(generator.next(value));
        } catch (e) {
          reject(e);
        }
      };
      var rejected = (value) => {
        try {
          step(generator.throw(value));
        } catch (e) {
          reject(e);
        }
      };
      var step = (x) =>
        x.done
          ? resolve(x.value)
          : Promise.resolve(x.value).then(fulfilled, rejected);
      step((generator = generator.apply(__this, __arguments)).next());
    });
  };

  // src/extension.js
  var loaderId = setInterval(() => {
    if (!window._gmailjs) {
      return;
    }
    clearInterval(loaderId);
    startExtension(window._gmailjs);
  }, 100);
  var extension_default = startExtension;
})();
//# sourceMappingURL=extension.js.map

function get_hook_url(desired_hook) {
  const urls = {
    corriger: "https://hook.eu1.make.com/8p42p5g0umgxs6g7dbnpecp4s77yvqqa",
    reformuler: "https://hook.eu1.make.com/o6mn1mo3h1ul2ax4o34bghmk1vv87kgf",
    reponse_tu: "https://hook.eu1.make.com/gj49o3ljyqfbpeqv569f3ok5elbkv94f",
    reponse_vous: "https://hook.eu1.make.com/1chjrnohmw9hsyevtou9ctptagkrqjes",
    traduction_fr_to_en:
      "https://hook.eu1.make.com/fqu1ayv9qd837rvm3gf6459nysyv4s22",
  };
  return urls[desired_hook];
}

async function startExtension(gmail) {
  window.gmail = gmail;
  gmail.observe.on("load", () => {
    gmail.observe.on("view_email", (domEmail) => {
      console.log("Looking at email:", domEmail);
      const emailData = gmail.new.get.email_data(domEmail);
      console.log("Email data:", emailData);
    });
    gmail.observe.on("compose", (compose) => {
      gmail.tools.add_compose_button(
        compose,
        "Corriger",
        async function () {
          const hook_url = get_hook_url("corriger");
          const data = get_email_data(compose);
          compose.body('Correction en cours...' + compose.body());
          const result = await process_data(data, hook_url);
          compose.body(result);
        },
        "Custom Style Classes"
      );
      gmail.tools.add_compose_button(
        compose,
        "Reformuler",
        async function () {
          const hook_url = get_hook_url("reformuler");
          const data = get_email_data(compose);
          compose.body("Reformulation en cours..." + compose.body());
          const result = await process_data(data, hook_url);
          compose.body(result);
        },
        "Custom Style Classes"
      );
      gmail.tools.add_compose_button(
        compose,
        "Reponse Tu",
        async function () {
          const hook_url = get_hook_url("reponse_tu");
          const data = get_email_data(compose);
          compose.body("Reponse en cours de calcul..." + compose.body());
          const result = await process_data(data, hook_url);
          compose.body(result);
        },
        "Custom Style Classes"
      );
      gmail.tools.add_compose_button(
        compose,
        "Reponse Vous",
        async function () {
          const hook_url = get_hook_url("reponse_vous");
          const data = get_email_data(compose);
          compose.body("Reponse en cours de calcul..." + compose.body());
          const result = await process_data(data, hook_url);
          compose.body(result);
        },
        "Custom Style Classes"
      );
      gmail.tools.add_compose_button(
        compose,
        "Traduction FR to EN",
        async function () {
          const hook_url = get_hook_url("traduction_fr_to_en");
          const data = get_email_data(compose);
          compose.body("Traduction en cours..." + compose.body());
          const result = await process_data(data, hook_url);
          compose.body(result);
        },
        "Custom Style Classes"
      );
    });
  });
}

function get_email_data(compose) {
  const currentEmailTo = compose.recipients();
  const currentEmailFrom = compose.from();
  const currentEmailBody = compose.body();
  const currentEmailSubject = compose.subject();
  const currentEmailId = compose.email_id();
  const currentThread = compose.thread_id();
  const threadDetails = gmail.new.get.thread_data(currentThread);
  let lastEmail = null;
  if (threadDetails) {
    for (let email of threadDetails.emails) {
      if (!email.is_draft) {
        lastEmail = email;
      }
      if (email.id === currentEmailId) {
        break;
      }
    }
  }
  const previousEmailBody = lastEmail ? lastEmail.body : "";
  const data = {
    emailTo: currentEmailTo,
    emailFrom: currentEmailFrom,
    currentEmailBody: prettify_text(remove_signature(currentEmailBody)),
    previousEmailBody: prettify_text(previousEmailBody),
    currentEmailSubject: currentEmailSubject,
  };
  return data;
}
```

```
function remove_signature(mail_body) {
  let current_email_body_without_signature = new DOMParser().parseFromString(
    mail_body,
    "text/html"
  );
  const signature = current_email_body_without_signature.querySelector(
    '[data-smartmail="gmail_signature"]'
  );
  if (signature) signature.remove();
  current_email_body_without_signature =
    current_email_body_without_signature.body.innerHTML;
  return current_email_body_without_signature;
}

function prettify_text(text) {
  if (!text) return "";
  let pretty_text = text.replace(/<br>/g, "\n");
  pretty_text = pretty_text.replace(/<\/p>/g, "\n");
  pretty_text = pretty_text.replace(/<p>/g, "");
  return pretty_text;
}

async function process_data(data, hook_url) {
  const output = await fetch_from_hook(data, hook_url);
  const pretty_output = prettify_text(output);
  return pretty_output;
}

async function fetch_from_hook(input, hook_url) {
  const response = await fetch(hook_url, {
    method: "POST",
    headers: {
      "Content-Type": "application/json",
    },
    body: JSON.stringify(input),
  });
  return response.text();
}
```

The remaining files comprise of .js and .json files that invoke different APIs and insert buttons into the browser. CSS was intended to be utilized for creating an enhanced interface.

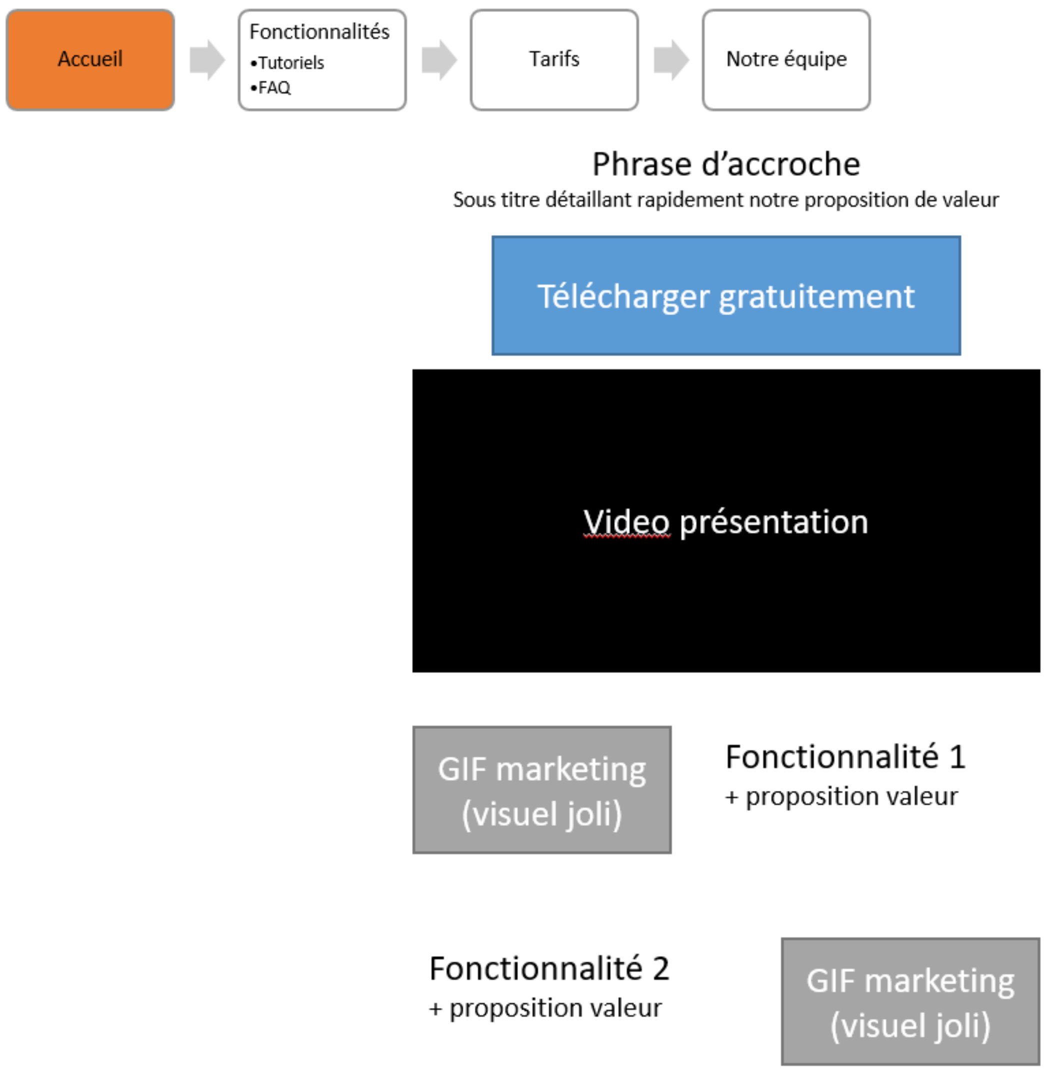
With Version 5.0, loading times have been significantly reduced (eliminating the need to wait for the currentemailbody to load as a draft), and the code has been optimized. Nevertheless, there are some encountered formatting issues and challenges in retrieving the "previousemailbody" context. As a result, further development has been put on hold, and the make.com scenarios have been deactivated. Blueprints can be provided upon request.

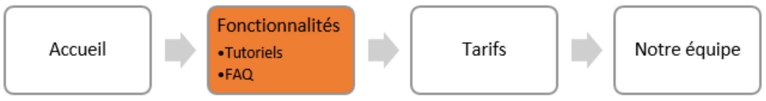
## Appendice 1 : Glossary

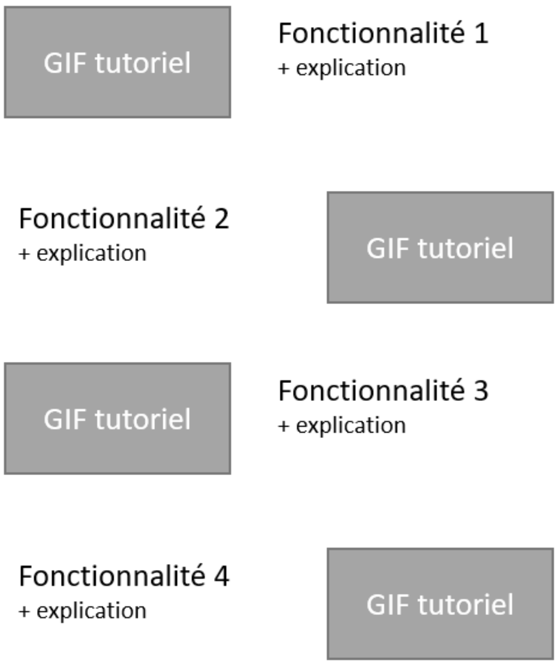
1. EasyAnswer: An intelligent tool designed to enhance Gmail by offering advanced features such as automatic correction, text rephrasing, predictive responses, and instant translation.
2. Gmail: A popular email service provided by Google.
3. Correction: The feature of EasyAnswer that identifies and suggests corrections for common spelling, grammar, and punctuation errors in email content.
4. Rephrasing: The capability of EasyAnswer to analyze written content and provide intelligent suggestions for improving clarity and conciseness.
5. Predictive responses: The functionality of EasyAnswer that generates pre-defined responses based on previous email history, allowing users to save time by selecting and customizing these responses instead of writing a complete reply each time.
6. Instant translation: The module integrated into EasyAnswer that enables users to select text in an email and choose a target language for quick and accurate translation, facilitating communication with individuals who speak different languages.
7. User productivity: The goal of EasyAnswer to improve email writing efficiency and enhance users' ability to effectively communicate through email.
8. Extension: Refers to the EasyAnswer tool, which extends the functionality of Gmail.
9. Stakeholders: Individuals or entities involved or interested in the development and use of EasyAnswer, such as users, developers, and potential collaborators.
10. Market analysis: A thorough study conducted to assess the commercial potential of EasyAnswer, including evaluating the email market and identifying integration and collaboration opportunities with other industry players.
11. System architecture: The clear structure and selection of appropriate technologies implemented in EasyAnswer to ensure performance, reliability, and security.

12. Development plan: A detailed plan followed during the development of EasyAnswer, including iterative iterations and rigorous testing to ensure software quality.
13. Results: The promising outcomes obtained from evaluating the effectiveness and added value of EasyAnswer for Gmail users, as well as the achievement of project objectives, system performance, and cost-benefit analysis.
14. APIs: Application Programming Interfaces used in EasyAnswer to interact with various functionalities and services.
15. Blueprints: Detailed plans or specifications of EasyAnswer's design and functionality, available upon request.
16. CSS: Cascading Style Sheets, a technology used for designing and improving the visual appearance and layout of web pages.
17. Version 5.0: The latest release of EasyAnswer, offering reduced loading times and optimized code, although encountering some formatting issues and difficulties in retrieving the "previousemailbody" context.
18. Suspended development: The temporary halt in further enhancements and updates to EasyAnswer.

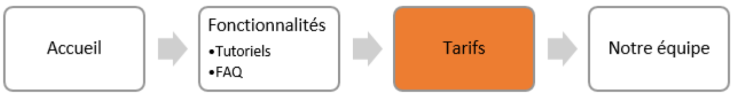
## Apppendice 2 : website prototype

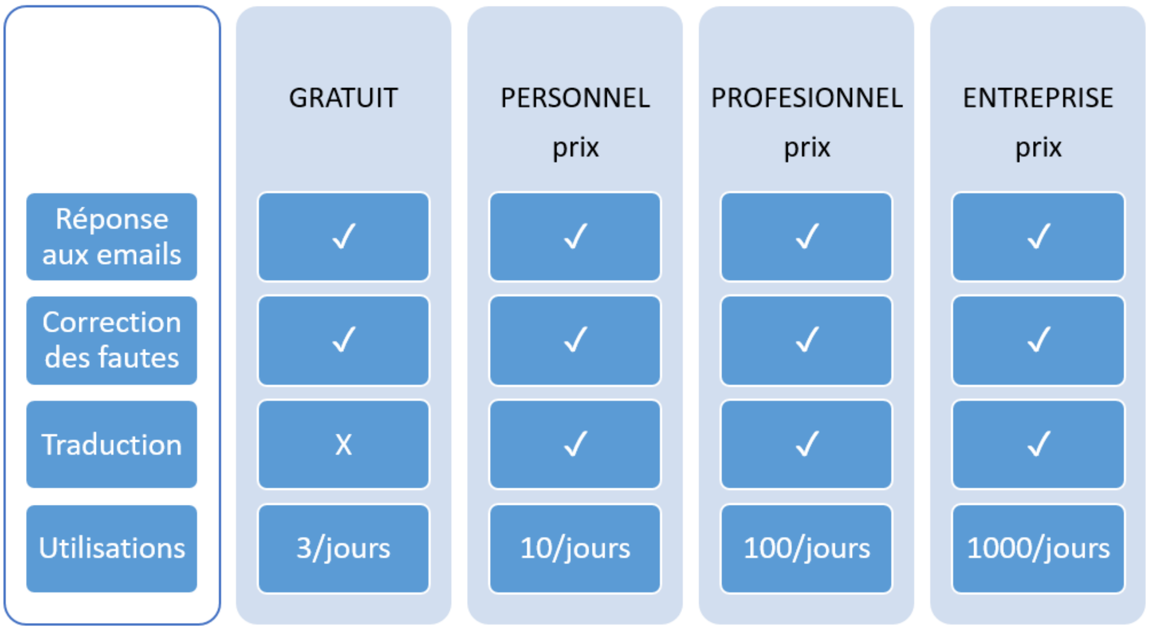






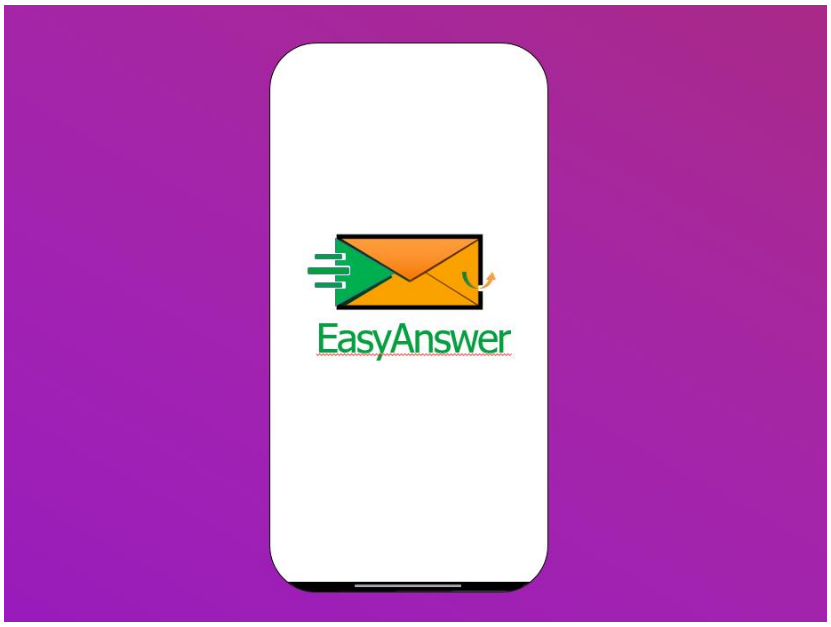
+2$^{e}$ onglet FAQ classique



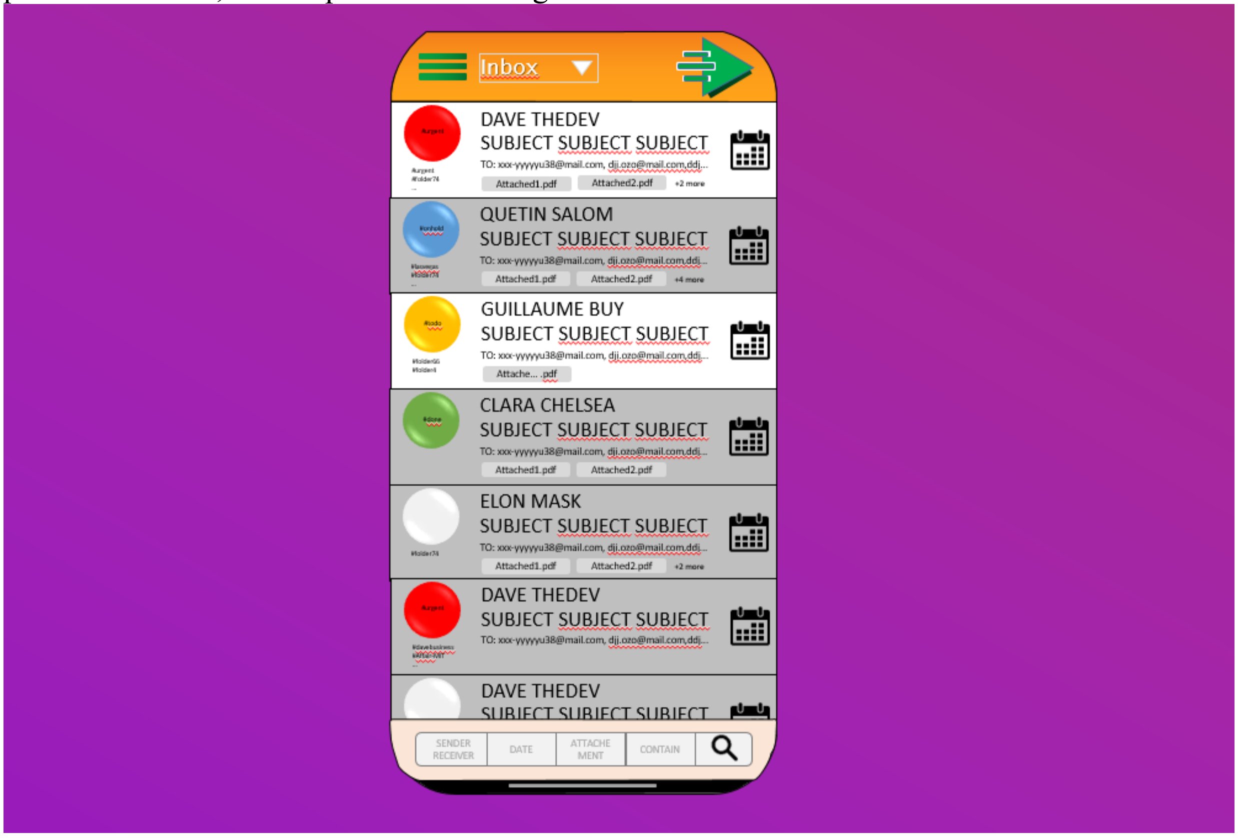


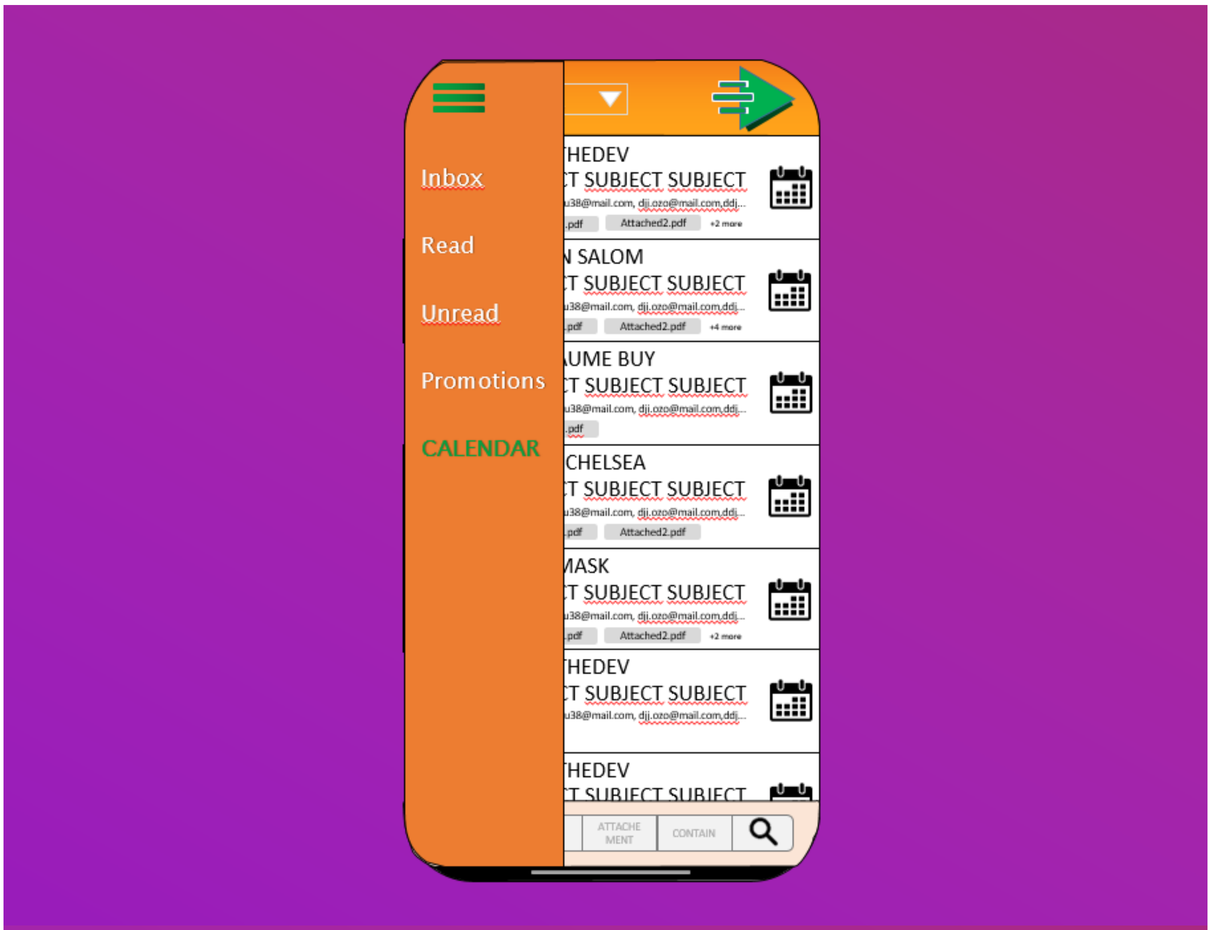
| | GRATUIT | PERSONNEL prix | PROFESIONNEL prix | ENTREPRISE prix |
|---|---|---|---|---|
| Réponse aux emails | ✓ | ✓ | ✓ | ✓ |
| Correction des fautes | ✓ | ✓ | ✓ | ✓ |
| Traduction | X | ✓ | ✓ | ✓ |
| Utilisations | 3/jours | 10/jours | 100/jours | 1000/jours |

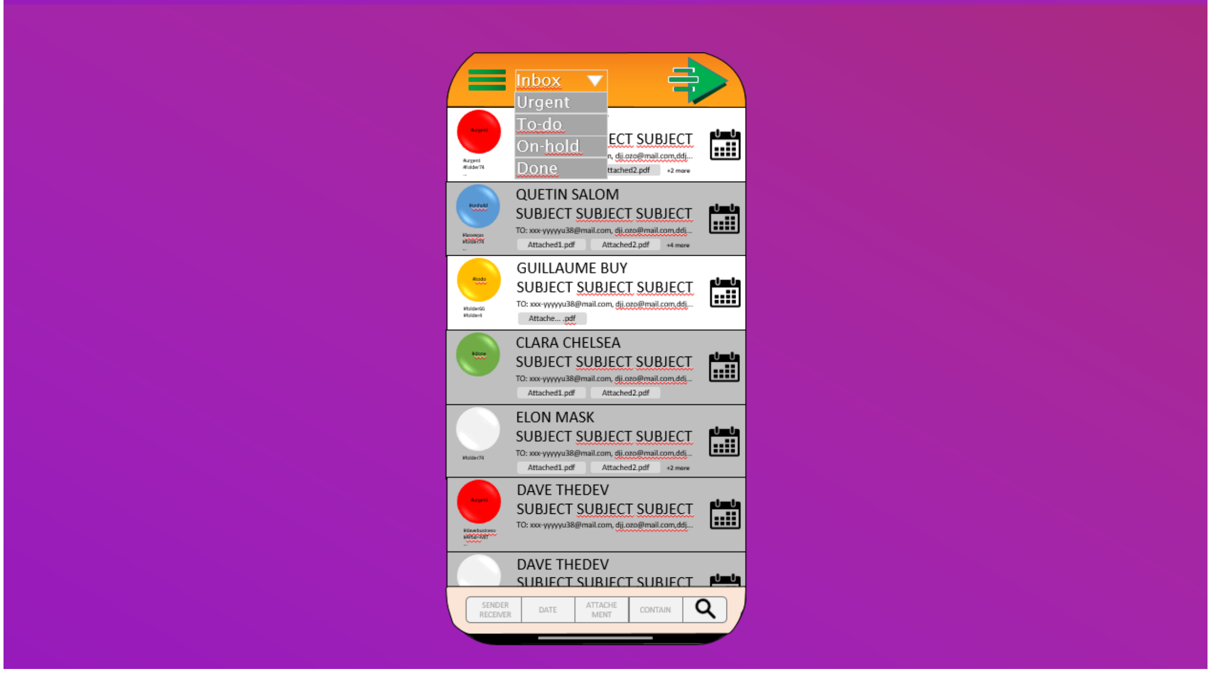
## Appendice 3 : mobile app prototype



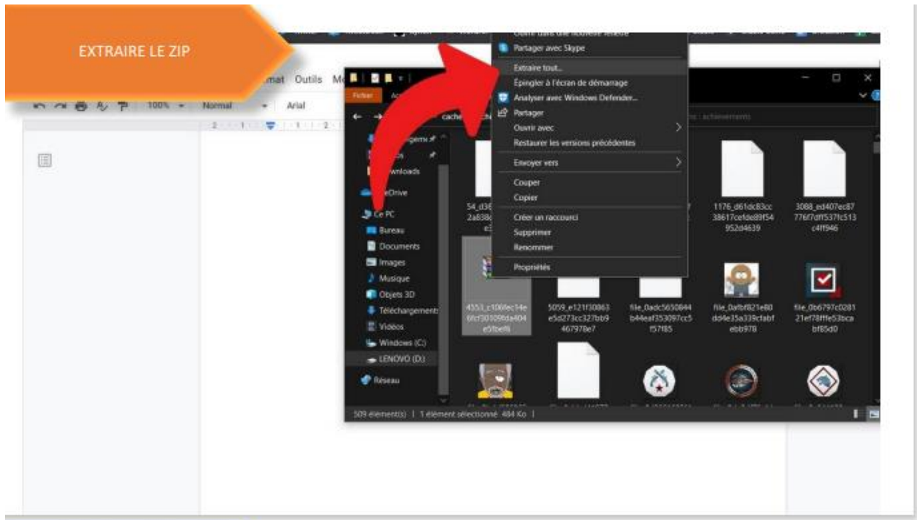
L'animation a été créée avec PowerPoint, et elle débute par un fondu en entrée de la lettre orange. Ensuite, la flèche de droite apparaît, suivie du logo EasyAnswer à gauche. Pendant ce temps, le logo EasyAnswer se déplace de manière dynamique jusqu'à prendre une position centrale, tandis que la lettre orange réalise un fondu de sortie.

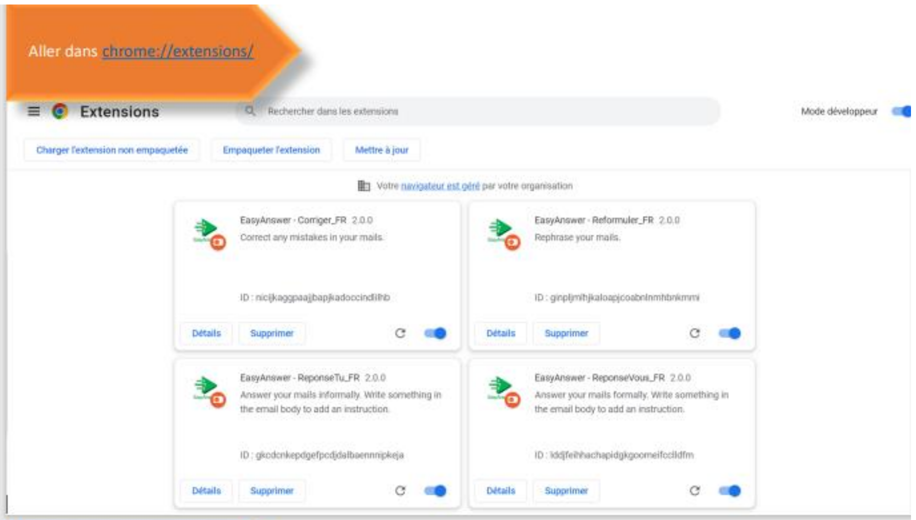



Inbox
Read
Unread
Promotions
CALENDAR
THEDEV
T SUBJECT SUBJECT
N SALOM
UME BUY
CHELSEA
MASK
ATTACHE MENT
CONTAIN


Inbox
Urgent
To-do
On-hold
Done
QUETIN SALOM
SUBJECT SUBJECT SUBJECT
GUILLAUME BUY
CLARA CHELSEA
ELON MASK
DAVE THEDEV
SENDER RECEIVER
DATE
ATTACHE MENT
CONTAIN

## Appendice 4 : Tutorial included in the files


EXTRAIRE LE ZIP


Aller dans chrome://extensions/

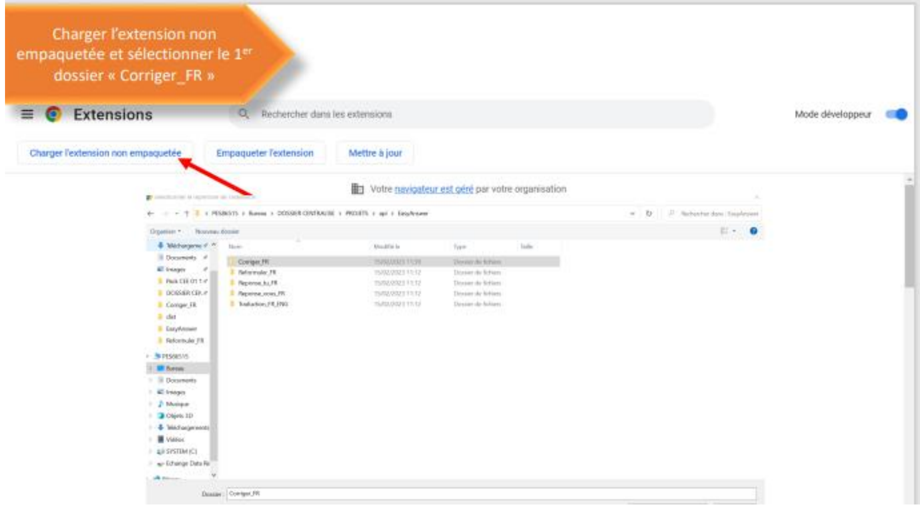
Extensions
Rechercher dans les extensions
Mode développeur
Charger l'extension non empaquetée
Empaqueter l'extension
Mettre à jour
Votre navigateur est géré par votre organisation

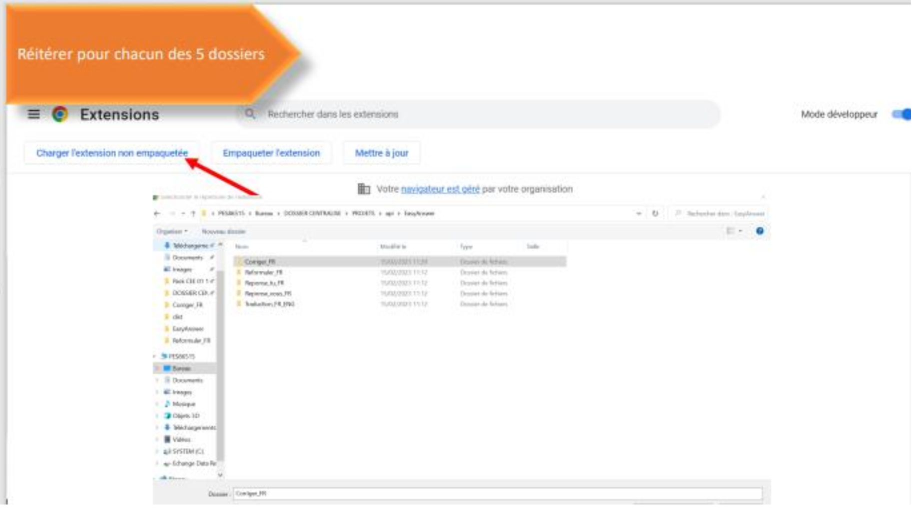
EasyAnswer - Corriger_FR 2.0.0
Correct any mistakes in your mails.
ID : nicijkaggpaajjbapjkadoccindlihb

EasyAnswer - Reformuler_FR 2.0.0
Rephrase your mails.
ID : ginpljmihjkaloapjcoabnlnmhbnkmmi

EasyAnswer - ReponseTu_FR 2.0.0
Answer your mails informally. Write something in the email body to add an instruction.
ID : gkcdcnkepdgefpcdjdalbaennnipkeja

EasyAnswer - ReponseVous_FR 2.0.0
Answer your mails formally. Write something in the email body to add an instruction.
ID : lddjfeihhachapidgkgoomeifoclldfm


Charger l'extension non empaquetée et sélectionner le 1er dossier « Corriger_FR »


Réitérer pour chacun des 5 dossiers

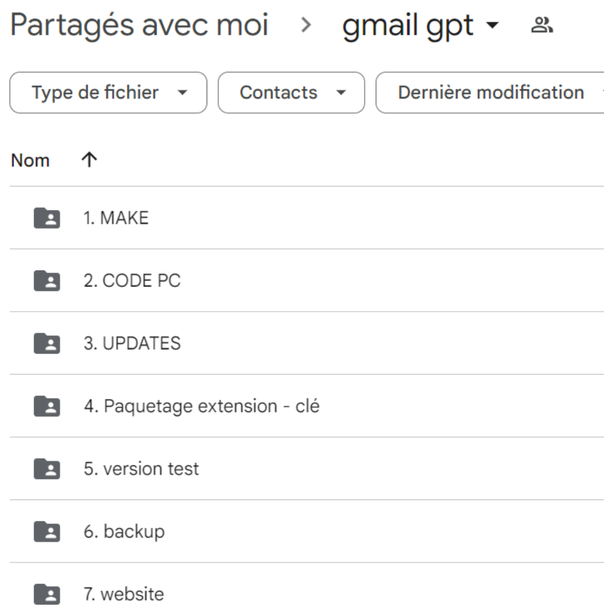
Extensions
Rechercher dans les extensions
Mode développeur
Charger l'extension non empaquetée
Empaqueter l'extension
Mettre à jour
Votre navigateur est géré par votre organisation
Dossier : Corriger_FR

## Appendice 5 : older brand image

## Appendice 6 : organisation

| TÂCHE | DESCRIPTION | ATTRIBUEE A | URGENCE (1 à 5) |
|---|---|---|---|
| Email Body | Corriger les bugs. Enquêter sur l'appli concurrente ? une extension permet de lire le code des extensions | Q&G | 5 |
| Protéger code | https://obfuscator.io/ pour commencer puis ? | Guillaume | 4 |
| Créer site | Français d'abord puis anglais. Onglet "pricing" | Quentin | 4 |
| Slogan | A définir | Q&G | 4 |
| ChatGPT | Rentrer les bonnes consignes et s'assurer du fonctionnement sur make.com | Quentin | 4 |
| Personnaliser+ optimiser code | github copilot pour comprendre puis ? | Guillaume | 3 |
| Abonnement | Q - créer sur la plateforme. Chosir tarif. G- Créer un compteur d'utilisations quotidiennes. Débloquer ce compteur une fois payé* | Q&G | 3 |
| paiement | coder nombre d'utilisations / jour + système verif clé | | 3 |
| Optimiser API | https://www.youtube.com/watch?v=bB7xkRsEq-g | G | 2 |
| Application | Android / Appstore : développer alternative à Gmail | | 2 |
| bouton | ajouter un bouton comme merlin pour envoyer vers le paiement | | 2 |
| Traduction | Appli en Anglais > Espagnol > Italien > Indonésien > Indien > Japonais https://financesonline.com/number-of-active-gmail-users/ | Q&G | 1 |
| Embauche | Trouver un dev pour tout intégrer en offline (sans make.com) | Quentin | 1 |

| | | | |
|---|---|---|---|
| Outlook | Convertir pour cette appli | | 1 |
| text loading | Mettre plusieurs options qui changent (boucle random) | Guillaume | 1 |
| anim loading | personnaliser avec notre logo ? | Guillaume | 1 |
| email entrant | option: traduire l'email reçu (sans lui répondre, juste pour le comprendre). 2e option: résumer email reçu | Q&G | 1 |
| MAJ | comment faire des MAJ avec chrome extension | | 0 |
| APK | possible de compiler notre extension en un apk ? | | 0 |
| délai 7s | Récupérer balise HTML au lieu de currentemail body pour résoudre | Guillaume | 5 |


Partagés avec moi > gmail gpt

1. MAKE
2. CODE PC
3. UPDATES
4. Paquetage extension - clé
5. version test
6. backup
7. website