Technical Documentation

Έκδοση: 1.0
Ημερομηνία: 28/08/2025
Συντάκτης: Πουλημένος Γεώργιος

1. Περιγραφή Συστήματος

Το **Athengen AI Interface** είναι μια web εφαρμογή βασισμένη σε Flask που:

- Διαβάζει emails/forms (.eml αρχεία)
- Εξάγει βασικά δεδομένα (Subject, From, To, Body)
- Επιτρέπει χειροκίνητη επεξεργασία μέσω browser
- Αποθηκεύει δεδομένα σε **Google Sheets**
- Χρησιμοποιεί μοντέλο **kriton/greek-text-summarization** για σύνοψη κειμένων

Η εφαρμογή λειτουργεί τοπικά με browser UI.

2. Τεχνολογίες

Κατηγορία	Τεχνολογία	Περιγραφή
Γλώσσα	Python 3.11.9	Backend ανάπτυξη
Γλώσσα	HTML / CSS	Frontend ανάπτυξη
Γλώσσα	Javascript	Fronted ανάπτυξη / Request/ Fetch

Framework	Flask	Web server & API
NLP	Hugging Face Transformers	Summarization μοντέλο
NLP Utils	spacy, sentence-transformers	Tokenization & embeddings
Δεδομένα	numpy, pandas, scikit-learn	Data handling
Storage	gspread, oauth2client	Σύνδεση με Google Sheets
Cloud API	Google Cloud API (Sheets, Drive)	Πρόσβαση και διαχείριση Google Sheets μέσω service account
Utilities	dill, pickle-mixin, requests, bs4	Βοηθητικές λειτουργίες
Metrics	rouge-score	Αξιολόγηση NLP outputs

3. Εγκατάσταση

Σε Windows :

```
#Download github repo
# unzip Document athengen_ai/
#Άνοιγμα τον φάκελο του project athengen_ai/
#Δεξί κλικ στο αρχείο  run_flask_app.bat
```

Σε Linux :

```
#Download github repo
# unzip Document athengen_ai/
#Άνοιγμα τον φάκελο του project athengen_ai/
#Άνοιγμα Command Line
# Εκτέλεση εντολής στο cmd : chmod +x run_flask_app.sh
# Εκτέλεση εντολής:  ./run_flask_app.sh
```

Διαφορετικά:

```
#Download github repo
# unzip Document athengen_ai/
#Άνοιγμα τον φάκελο του project athengen_ai/
#Άνοιγμα Command Line
# Εκτέλεση εντολής στο cmd : pip install request
# Εκτέλεση εντολής:  .python run.py
```

4. Δομή Αρχείων

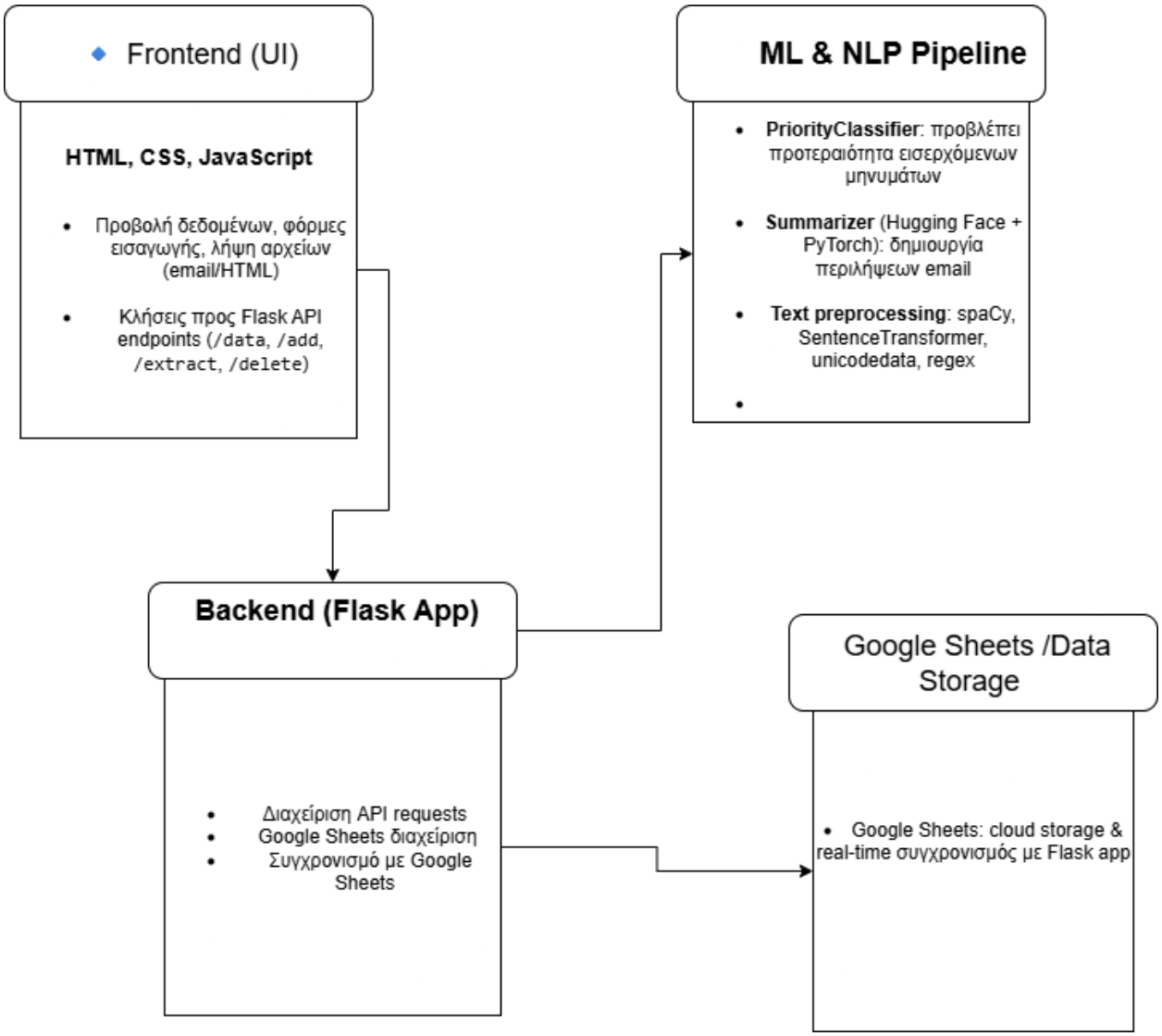
```
athengen_ai/
├──config.py
├── app.py              # Flask entry point
├── templates/          # HTML templates
├── emails/             #Φάκελος με dummy data
├── routes/
│   ├── routes.py       # Διαχείριση Endpoints
│
├── static/             # CSS, JS
│   ├── style.css
│   └── app.js
├── services/
│   ├── data_services.py            #Επικοινωνία με το Google Sheet
│   └── extractor.py                #Email extractor
│   └── form_extractor.py           #Form Extractor
├── auto_testing.py           #Unit Tests
├──credential.json            #Απαραίτητο για σύνδεση με Google Cloud API
├── priority_classifier.py     # Ταξινομητής (Εκπαίδευση -Κώδικας)
├── priority_model.pk          # Μοντέλο Ταξινομητή
├── run_flask.sh              #Εκτελέσιμο πρόγραμμα για Linux
├── run_flask.bat             # Εκτελέσιμο Πρόγραμμα Windows
├── Ανάλυση Αναγκών-Τεχνική Πρόταση.pdf
├── Technical Documentation.pdf
├── User Manual.pdf
└── README.md
```

5. Endpoints (API)

Endpoint	Μέθοδος	Περιγραφή
`/`	GET	Εμφανίζει την αρχική σελίδα (`index.html`).
`/data`	GET	Επιστρέφει όλα τα δεδομένα από το Google Sheet σε μορφή JSON.
`/add`	POST	Προσθέτει μια νέα γραμμή στο Google Sheet.
`/delete_by_source/`	DELETE	Διαγράφει όλες τις γραμμές με κενό `Source`.
`/delete_by_source/<source>`	DELETE	Διαγράφει όλες τις γραμμές που έχουν συγκεκριμένο `Source`.
`/extract/<filename>`	POST	Κάνει extract δεδομένα από αρχείο `.eml` ή `.html` και τα αποθηκεύει στο Google Sheet.
`/list_emails`	GET	Επιστρέφει λίστα με όλα τα `.eml` αρχεία από τον φάκελο `EMAIL_FOLDER`.
`/list_forms`	GET	Επιστρέφει λίστα με όλα τα `.html` αρχεία από τον φάκελο `EMAIL_FOLDER`.
`/delete_file/<filename>`	DELETE	Διαγράφει αρχείο από τον τοπικό φάκελο `EMAIL_FOLDER`.
`/get_file_data/<filename>`	GET	Επιστρέφει δεδομένα που σχετίζονται με ένα αρχείο (από Google Sheet ή extract preview).
`/save_file_data`	POST	Αποθηκεύει χειροκίνητα επεξεργασμένα δεδομένα στο Google Sheet.

6.Αρχιτεκτονική Συστήματος

Η εφαρμογή αποτελείται από τέσσερα βασικά υποσυστήματα που συνεργάζονται:



1. **Frontend (UI)**
    - Υλοποιημένο σε **HTML, CSS, JavaScript**
    - Παρέχει φόρμες εισαγωγής, προβολή δεδομένων και υποβολή αρχείων
    - Καλεί API endpoints του Flask backend (`/data`, `/add`, `/extract`, `/delete`)
2. **Backend (Flask App)**
    - Διαχειρίζεται τα API requests από το UI

- Συνδέεται με το Google Sheets API για ανάγνωση/εγγραφή
- Συντονίζει τα modules του ML/NLP pipeline

3. **ML & NLP Pipeline**
   - **PriorityClassifier**: Πρόβλεψη προτεραιότητας εισερχόμενων μηνυμάτων
   - **Summarizer**: Δημιουργία περιλήψεων (HuggingFace + PyTorch)
   - **Text preprocessing**: spaCy, SentenceTransformer, regex, unicodedata
4. **Google Sheets / Data Storage**
   - Χρησιμοποιείται ως αποθηκευτικός χώρος στο cloud
   - Παρέχει **real-time συγχρονισμό** με το Flask backend

Η ροή δεδομένων είναι:
User (UI) → Flask App → ML/NLP Pipeline → Google Sheets → Επιστροφή στο UI

7.Use Case Scenarios

Use Case 1: Εξαγωγή δεδομένων από email

Σενάριο:
Ο χρήστης θέλει να εξάγει πληροφορίες από ένα αρχείο email (.eml) και να τις αποθηκεύσει στο Google Sheet.

Βήματα:

1. Ο χρήστης ανεβάζει το αρχείο .eml μέσω του browser UI.
2. Το frontend καλεί το endpoint `/extract/<filename>` με μέθοδο POST.
3. Το backend διαβάζει το email και εξάγει: Subject, From, To, Body.
4. Ο χρήστης βλέπει preview των δεδομένων στην οθόνη.
5. Ο χρήστης μπορεί να επεξεργαστεί χειροκίνητα τα δεδομένα.

6. Πατώντας “Save”, τα δεδομένα αποθηκεύονται στο Google Sheet μέσω του endpoint `/save_file_data`.

Εναλλακτικά Σενάρια:

- Αν το αρχείο δεν είναι έγκυρο (.eml ή .html), εμφανίζεται μήνυμα σφάλματος.
- Αν η σύνδεση με το Google Sheet αποτύχει, τα δεδομένα δεν θα αποθηκευτούν - μήνυμα σφάλματος

Use Case 2: Προβολή όλων των emails ή forms για επεξεργασία

Σενάριο:
Ο χρήστης θέλει να δει ποια αρχεία emails και forms είναι διαθέσιμα για extract.

Βήματα:

1. Ο χρήστης ανοίγει το UI στην ενότητα “List Files”.
2. Το frontend καλεί τα endpoints `/list_emails` και `/list_forms`.
3. Το backend επιστρέφει λίστες με τα διαθέσιμα αρχεία.
4. Ο χρήστης βλέπει τα ονόματα των αρχείων και επιλέγει ποιο θέλει να επεξεργαστεί ή να εξάγει.

Use Case 3: Διαγραφή αρχείων από το inbox

Σενάριο:
Ο χρήστης θέλει να διαγράψει συγκεκριμένα emails ή forms ή γραμμές από το Google Sheet.

Βήματα:

1. Ο χρήστης επιλέγει αρχεία από το UI.
2. Για διαγραφή αρχείων, το frontend καλεί `/delete_file/<filename>` (DELETE).
3. Για διαγραφή δεδομένων από το Google Sheet, καλεί `/delete_by_source/<source>` (DELETE) ή `/delete_by_source/` για κενές

εγγραφές.

4. Το backend διαγράφει τα αρχεία/γραμμές και επιστρέφει επιβεβαίωση.

Εναλλακτικά Σενάρια:

- Αν το αρχείο δεν υπάρχει, επιστρέφεται σφάλμα.
- Αν το Google Sheet δεν είναι διαθέσιμο, εμφανίζεται μήνυμα αποτυχίας.

Use Case 4: Αυτόματη δημιουργία περιλήψεων (Summarization) των email των πελατών

Σενάριο:
Ο χρήστης θέλει να δει συνοπτική εκδοχή ενός email ή form.

Βήματα:

1. Ο χρήστης επιλέγει ένα αρχείο για εξαγωγή
2. Το frontend καλεί το endpoint `/extract/<filename>`
3. Το backend χρησιμοποιεί το μοντέλο `kriton/greek-text-summarization` για να δημιουργήσει περίληψη για το message.
4. Η περίληψη των email εμφανίζεται στο UI μαζί με τις υπόλοιπες πληροφορίες που εξαγονται απο το email, όπου ο χρήστης μπορεί να την επεξεργαστεί πρωτού να την αποθηκεύσει στο Google Sheet και να εμφανιστει στο table του UI..

Use Case 5: Προτεραιοποίηση μηνυμάτων

Σενάριο:
Ο σύστημα τα εισερχόμενα email κατά προτεραιότητα.

Βήματα:

1. Το σύστημα καλεί το ML module `PriorityClassifier` για κάθε εισερχόμενο μήνυμα, όταν ενεργοποιηθεί η λειτουργία του Εξαγωγής δεδομένων .

2. Η πρόβλεψη προτεραιότητας αποθηκεύεται στο Google Sheet μαζί με τα υπόλοιπα δεδομένα.

3. Ο χρήστης μπορεί να φιλτράρει και να εμφανίσει μηνύματα υψηλής ή χαμηλής προτεραιότητας μέσω του UI.

Use Case 6: Αναζήτηση δεδομένων στον πίνακα

Σενάριο:
Ο χρήστης θέλει να βρει συγκεκριμένα emails ή forms στον πίνακα με βάση λέξεις-κλειδιά.

Βήματα:

1. Ο χρήστης εισάγει λέξη-κλειδί ή φράση στο UI πεδίο αναζήτησης.

2. Το frontend στέλνει αίτημα στο backend για αναζήτηση στα δεδομένα του Google Sheet.

3. Το backend επιστρέφει όλες τις γραμμές που περιέχουν τη λέξη-κλειδί σε Subject, From, Body ή άλλες στήλες.

4. Το UI εμφανίζει τα αποτελέσματα στον χρήστη.

Εναλλακτικά Σενάρια:

- Αν δεν βρεθεί καμία εγγραφή, εμφανίζεται μήνυμα “No results found”.

Use Case 7: Φιλτράρισμα δεδομένων

Σενάριο:
Ο χρήστης θέλει να φιλτράρει emails ή forms με βάση συγκεκριμένα κριτήρια, όπως προτεραιότητα, τύπο αρχείου ή ημερομηνία.

Βήματα:

1. Ο χρήστης πληκτρολογεί οποιοδήποτε χαρακτηριστικό ή έστω έναν αριθμό από οποιαδήποτε χαρακτηριστικό π.χ. αν θέλω email γράφω “E” ή “‘Email” ή “Em” κλπ.

   ή π.χ. high αν θέλουμε αντικείμενα του πινακα με high Priority

2. Το frontend καλεί το backend για φιλτράρισμα στον πίνακα.
3. Το backend εφαρμόζει τα φίλτρα και επιστρέφει τα αποτελέσματα.
4. Το UI εμφανίζει μόνο τις γραμμές που πληρούν τα κριτήρια.

Use Case 8: Διαγραφή δεδομένων από τον πίνακα

Σενάριο:
Ο χρήστης θέλει να διαγράψει μία ή περισσότερες γραμμές από το Google Sheet.

Βήματα:

1. Ο χρήστης επιλέγει γραμμή στο UI (πατάει το κουμπί “Διαγραφής”).
2. Το frontend στέλνει αίτημα DELETE στο endpoint `/delete_by_source/<source>` ή σε συγκεκριμένες γραμμές.
3. Το backend διαγράφει τις γραμμές από το Google Sheet και επιστρέφει επιβεβαίωση.

Εναλλακτικά Σενάρια:

- Αν το Google Sheet δεν είναι διαθέσιμο, εμφανίζεται μήνυμα σφάλματος.

Use Case 9: Διαχείριση email / invoice

Σενάριο:
Ο χρήστης θέλει να ξεχωρίσει και να επεξεργαστεί emails ή invoices που περιέχονται στον πίνακα.

Βήματα:

1. Το σύστημα ταξινομεί αρχεία κατά τύπο (email/contact, invoice) μέσω μιας λίστας επιλογής.

Use Case 10: Επεξεργασία δεδομένων απευθείας από το Google Sheet

Σενάριο:
Ο χρήστης θέλει να επεξεργαστεί δεδομένα που έχουν ήδη εξαχθεί (extract) και βρίσκονται στο Google Sheet, χωρίς να χρειάζεται να ξανακάνει extract ή να αλλάξει τα αρχεία στον τοπικό φάκελο.

Πρωταρχικά βήματα:

1. Ο χρήστης ανοίγει το UI και βλέπει τον πίνακα με όλα τα δεδομένα που έχουν αποθηκευτεί στο Google Sheet.

2. Ο χρήστης επιλέγει μία ή περισσότερες γραμμές για επεξεργασία.

3. Ο χρήστης τροποποιεί πεδία όπως **Subject, From, To, Body, Amount, Date**, ανάλογα με τον τύπο του αρχείου (email ή form).

4. Μετά την επεξεργασία, ο χρήστης πατάει “Save”.

5. Το frontend καλεί το endpoint `/save_file_data` με μέθοδο POST.

6. Το backend ενημερώνει το Google Sheet με τις αλλαγές και επιστρέφει επιβεβαίωση επιτυχίας.

Εναλλακτικά Σενάρια:

- Αν η σύνδεση με το Google Sheet αποτύχει, εμφανίζεται μήνυμα σφάλματος και τα δεδομένα δεν ενημερώνονται.

- Αν τα δεδομένα έχουν τροποποιηθεί παράλληλα από άλλον χρήστη, μπορεί να εμφανιστεί προειδοποίηση σύγκρουσης (conflict).

Σημείωση:

- Αυτή η διαδικασία αφορά μόνο δεδομένα που ήδη υπάρχουν στο Google Sheet και **δεν επηρεάζει τα πρωτότυπα αρχεία (.eml, .html)** στον τοπικό φάκελο.

- Το UI μπορεί να παρέχει λειτουργίες **inline edit** ή pop-up φόρμες για ευκολότερη επεξεργασία.

8. Performance & Limitations

1 Μέγιστος αριθμός αρχείων που μπορούν να ανεβούν ταυτόχρονα

- Το σύστημα μπορεί να επεξεργαστεί **μέχρι 5-10 αρχεία ταυτόχρονα** μέσω του UI, ανάλογα με το μέγεθος των αρχείων.

2 Χρονικά όρια για extract & summarization

- **Extract:** περίπου 1-2 δευτερόλεπτα ανά email (.eml) ή form (.html) μικρού μεγέθους (<100 KB).

- **Summarization:** 2-5 δευτερόλεπτα ανά κείμενο, ανάλογα με το μέγεθος του body και την πολυπλοκότητα του κειμένου.

- Σε μεγάλα αρχεία (>1 MB), η επεξεργασία μπορεί να καθυστερήσει σημαντικά.

3 Εξάρτηση από τα όρια του Google Sheets API

- Το Google Sheets API έχει **όριο 500 requests ανά 100 δευτερόλεπτα ανά project** (rate limit).

- Για μεγάλες μαζικές ενημερώσεις, συνιστάται:

    - Χρήση **batch updates** αντί για πολλά μικρά requests

- Αποτυχία λόγω υπέρβασης του rate limit εμφανίζει **HTTP 429 Too Many Requests**, οπότε πρέπει να γίνεται retry με delay.