# HR\_Management Developer Manual

Generated: 2025-10-15T13:57:47.717720+07:00

## Table of Contents

1. hr\_management\_app\database\\_\_init\_\_.py
2. hr\_management\_app\ml\\_\_init\_\_.py
3. hr\_management\_app\ml.py
4. hr\_management\_app\parsers\\_\_init\_\_.py
5. hr\_management\_app\parsers.py
6. hr\_management\_app\scripts\send\_test\_email.py
7. hr\_management\_app\send\_test\_fpt.py
8. hr\_management\_app\send\_test\_tmp.py
9. hr\_management\_app\src\\_\_init\_\_.py
10. hr\_management\_app\src\account\_manage.py
11. hr\_management\_app\src\admin\_author.py
12. hr\_management\_app\src\auth\_gui.py
13. hr\_management\_app\src\conftest.py
14. hr\_management\_app\src\contracts\\_\_init\_\_.py
15. hr\_management\_app\src\contracts\gui\_subsets.py
16. hr\_management\_app\src\contracts\models.py
17. hr\_management\_app\src\contracts\views.py
18. hr\_management\_app\src\database\\_\_init\_\_.py
19. hr\_management\_app\src\database\database.py
20. hr\_management\_app\src\email\_config.py
21. hr\_management\_app\src\employees\\_\_init\_\_.py
22. hr\_management\_app\src\employees\models.py
23. hr\_management\_app\src\gui.py
24. hr\_management\_app\src\main.py
25. hr\_management\_app\src\ml\\_\_init\_\_.py
26. hr\_management\_app\src\ml\imputer.py
27. hr\_management\_app\src\ml\imputer\_heuristic.py
28. hr\_management\_app\src\ml\imputer\_ml.py
29. hr\_management\_app\src\ml\ner.py
30. hr\_management\_app\src\ml\train\_spacy\_ner.py
31. hr\_management\_app\src\parsers\\_\_init\_\_.py
32. hr\_management\_app\src\parsers\file\_parser.py
33. hr\_management\_app\src\parsers\mapping\_store.py
34. hr\_management\_app\src\parsers\normalizer.py
35. hr\_management\_app\src\smoke\_test.py
36. hr\_management\_app\src\test\_import\_integration.py
37. hr\_management\_app\src\test\_smtp.py
38. hr\_management\_app\src\test\_ui\_import\_dialog.py
39. hr\_management\_app\src\tests\test\_contracts.py
40. hr\_management\_app\src\tests\test\_contracts\_colors.py
41. hr\_management\_app\src\tests\test\_contracts\_soft\_delete.py
42. hr\_management\_app\src\tests\test\_database.py
43. hr\_management\_app\src\tests\test\_ml\_ner.py
44. hr\_management\_app\src\tests\test\_normalizer.py
45. hr\_management\_app\src\tests\test\_parsers.py
46. hr\_management\_app\src\tests\test\_role\_audit\_and\_permissions.py
47. hr\_management\_app\src\tests\test\_search\_contracts\_and\_employees.py
48. hr\_management\_app\src\tests\test\_user\_management.py
49. hr\_management\_app\src\tests\test\_validators.py
50. hr\_management\_app\src\ui\_helpers.py
51. hr\_management\_app\src\ui\_import.py
52. hr\_management\_app\src\ui\_validators.py
53. hr\_management\_app\tests\test\_contract\_trash.py
54. hr\_management\_app\tests\test\_imputation\_audit.py
55. hr\_management\_app\tests\test\_integration\_import\_flow.py
56. hr\_management\_app\tests\test\_ml\_imputer.py
57. hr\_management\_app\tests\test\_parsers\_csv.py
58. hr\_management\_app\tests\test\_pending\_contracts\_gui\_integration.py
59. hr\_management\_app\tests\test\_pending\_contracts\_pytest.py
60. hr\_management\_app\tmp\_check\_test.py
61. hr\_management\_app\tmp\_gui\_smoke\_sim.py
62. hr\_management\_app\tmp\_pending\_contract\_smoke.py
63. hr\_management\_app\tools\apply\_heuristic\_imputer.py
64. hr\_management\_app\tools\check\_sklearn\_import.py
65. hr\_management\_app\tools\force\_train\_sklearn.py
66. hr\_management\_app\tools\generate\_dummy\_20k.py
67. hr\_management\_app\tools\generate\_dummy\_xlsx.py
68. hr\_management\_app\tools\launch\_import\_gui.py
69. hr\_management\_app\tools\preview\_imputations\_headless.py
70. hr\_management\_app\tools\run\_import\_xlsx.py
71. hr\_management\_app\tools\run\_smoke\_impute.py
72. hr\_management\_app\tools\smoke\_imports.py
73. hr\_management\_app\tools\smoke\_impute.py
74. hr\_management\_app\tools\smoke\_test\_imputer.py
75. hr\_management\_app\tools\train\_imputer.py
76. hr\_management\_app\tools\train\_imputer\_from\_xlsx.py

## hr\_management\_app\database\\_\_init\_\_.py

Forwarding package `database` that re-exports src.database.database.

## hr\_management\_app\ml\\_\_init\_\_.py

Forwarding package `ml` that re-exports implementations under src.ml.

## hr\_management\_app\ml.py

Top-level import shim for `ml` used by tools and scripts.

### \_\_getattr\_\_

def \_\_getattr\_\_(name)

Summary:

No summary available.

## hr\_management\_app\parsers\\_\_init\_\_.py

Forwarding package `parsers` that re-exports the implementation under src.parsers.  
This helps tools and static analysis resolve imports when running from the repo layout.

## hr\_management\_app\parsers.py

Top-level import shim for `parsers` used by tools and scripts.  
This forwards to the package under src if available.

### \_\_getattr\_\_

def \_\_getattr\_\_(name)

Summary:

No summary available.

## hr\_management\_app\scripts\send\_test\_email.py

Utility to send a test email using the app's send\_email helper.  
  
Usage (PowerShell):  
 $env:SMTP\_SERVER='smtp.example.com'; $env:SMTP\_PORT='587'; $env:SMTP\_USER='you@example.com'; $env:SMTP\_PASSWORD='app-pass'; $env:FROM\_EMAIL='you@example.com'  
 python ./scripts/send\_test\_email.py datnhism@gmail.com  
  
This script adds the package `src` to sys.path so it can be executed from the repo root.  
It logs full exceptions to the log and prints a friendly message to the console.

### main

def main()

Summary:

No summary available.

Side-effects: Sends email (side-effect)

## hr\_management\_app\send\_test\_fpt.py

No file-level docstring available.

## hr\_management\_app\send\_test\_tmp.py

No file-level docstring available.

## hr\_management\_app\src\\_\_init\_\_.py

No file-level docstring available.

## hr\_management\_app\src\account\_manage.py

No file-level docstring available.

### UserManagementWindow

class UserManagementWindow(tk.Toplevel)

Summary:

No summary available.

## hr\_management\_app\src\admin\_author.py

No file-level docstring available.

### AdminAuthPage

class AdminAuthPage(tk.Toplevel)

Summary:

No summary available.

## hr\_management\_app\src\auth\_gui.py

No file-level docstring available.

### SignUpWindow

class SignUpWindow(tk.Toplevel)

Summary:

No summary available.

### AuthWindow

class AuthWindow(tk.Tk)

Summary:

No summary available.

## hr\_management\_app\src\conftest.py

No file-level docstring available.

### session\_db\_setup

def session\_db\_setup()

Docstring:

Ensure schema exists for the session and cleanup at the end.

Summary:

Ensure schema exists for the session and cleanup at the end.

Side-effects: Performs file I/O or opens files

### clean\_tables\_before\_test

def clean\_tables\_before\_test()

Docstring:

Truncate key tables before each test to provide isolation while reusing the session DB.  
  
This is faster than creating a fresh DB file per test but avoids state leakage that  
caused unique-email collisions when tests create users.

Summary:

Truncate key tables before each test to provide isolation while reusing the session DB.

Side-effects: Modifies database

## hr\_management\_app\src\contracts\\_\_init\_\_.py

No file-level docstring available.

## hr\_management\_app\src\contracts\gui\_subsets.py

No file-level docstring available.

### status\_choices

def status\_choices()

Summary:

No summary available.

### show\_contract\_subsets

def show\_contract\_subsets(contract\_id, actor\_user\_id)

Summary:

No summary available.

## hr\_management\_app\src\contracts\models.py

No file-level docstring available.

### \_contract\_storage\_dir

def \_contract\_storage\_dir()

Summary:

No summary available.

### store\_contract\_file

def store\_contract\_file(src\_path, construction\_id)

Docstring:

Copy the given file into the project's contract storage and return the relative path.  
  
The returned path is absolute. Filenames are sanitized by prefixing employee id and a timestamp.

Summary:

Copy the given file into the project's contract storage and return the relative path.

Side-effects: Performs file I/O or opens files

### Contract

class Contract

Summary:

No summary available.

### Subset

class Subset

Summary:

No summary available.

### get\_subsets

def get\_subsets(contract\_id)

Summary:

No summary available.

### contract\_progress

def contract\_progress(contract\_id)

Docstring:

Return a small progress report for a contract based on subset statuses.  
  
- percent\_complete: fraction of subsets in COMPLETED\_STATUSES  
- total: number of subsets  
- completed: number completed  
- details: list of subset dicts

Summary:

Return a small progress report for a contract based on subset statuses.

## hr\_management\_app\src\contracts\views.py

No file-level docstring available.

### display\_contracts

def display\_contracts(contracts)

Summary:

No summary available.

### view\_contract

def view\_contract(contract)

Summary:

No summary available.

## hr\_management\_app\src\database\\_\_init\_\_.py

No file-level docstring available.

## hr\_management\_app\src\database\database.py

No file-level docstring available.

Deprecation / Hotspots:

Line 1057: fname = f"{datetime.utcnow().strftime('%Y%m%dT%H%M%SZ')}\_{uuid.uuid4().hex}.eml"

Line 1153: created\_at = datetime.utcnow().isoformat()

Line 1170: (datetime.utcnow().isoformat(), outbox\_id),

Line 1180: (last\_error, datetime.utcnow().isoformat(), outbox\_id),

Line 1588: submitted\_at = datetime.utcnow().isoformat()

Line 1643: approved\_at = datetime.utcnow().isoformat()

Line 1660: rejected\_at = datetime.utcnow().isoformat()

### \_conn

def \_conn()

Docstring:

Context manager that yields a sqlite3.Connection and ensures it is closed.  
  
Use like: with \_conn() as conn: ...  
This prevents ResourceWarnings when callers forget to close connections.

Summary:

Context manager that yields a sqlite3.Connection and ensures it is closed.

### init\_db

def init\_db()

Docstring:

Create required tables if missing.

Summary:

Create required tables if missing.

### \_create\_tables

def \_create\_tables(conn)

Docstring:

Create core tables on the given sqlite3 connection object.

Summary:

Create core tables on the given sqlite3 connection object.

Side-effects: Modifies database

### record\_imputation\_audit

def record\_imputation\_audit(row\_index, field, old\_value, new\_value, source, actor\_user\_id)

Docstring:

Record an imputation decision into the imputation\_audit table.  
  
row\_index: the index in the imported batch (0-based) for traceability.  
field: the field name that was imputed (e.g., 'job\_title').  
old\_value/new\_value: textual values prior to and after imputation.  
source: where the imputation originated (preview, ml, heuristic, etc.).  
actor\_user\_id: optional user id who accepted the imputation.

Summary:

Record an imputation decision into the imputation\_audit table.

Side-effects: Modifies database

### export\_imputation\_audit\_csv

def export\_imputation\_audit\_csv(path)

Docstring:

Export imputation\_audit table to CSV. Returns number of rows exported.

Summary:

Export imputation\_audit table to CSV. Returns number of rows exported.

Side-effects: Performs file I/O or opens files

### add\_contract\_to\_db

def add\_contract\_to\_db(contract)

Summary:

No summary available.

### get\_all\_contracts

def get\_all\_contracts()

Summary:

No summary available.

### get\_all\_contracts\_filtered

def get\_all\_contracts\_filtered(include\_deleted)

Docstring:

Return contracts rows; by default exclude soft-deleted rows.  
  
If include\_deleted is True, return all rows including deleted ones.  
The returned row shape prefers the full set of columns (including deleted/deleted\_at) when available.

Summary:

Return contracts rows; by default exclude soft-deleted rows.

### search\_contracts

def search\_contracts(term, include\_deleted, limit)

Docstring:

Search contracts by id, area, incharge, terms, or construction\_id.  
  
term: search string; numeric strings will be matched against id and construction\_id.  
include\_deleted: include soft-deleted rows when True.  
limit: optional cap on returned rows.  
Returns list of rows in the same shape as get\_all\_contracts\_filtered.

Summary:

Search contracts by id, area, incharge, terms, or construction\_id.

### create\_contract\_subset

def create\_contract\_subset(contract\_id, title, description, status, order\_index)

Summary:

No summary available.

Side-effects: Modifies database

### get\_subsets\_for\_contract

def get\_subsets\_for\_contract(contract\_id)

Summary:

No summary available.

### get\_child\_contracts

def get\_child\_contracts(contract\_id)

Docstring:

Return list of contracts that have parent\_contract\_id == contract\_id.

Summary:

Return list of contracts that have parent\_contract\_id == contract\_id.

### get\_subsets\_count

def get\_subsets\_count(contract\_id)

Summary:

No summary available.

### get\_contract\_by\_id

def get\_contract\_by\_id(contract\_id, include\_deleted)

Docstring:

Return a single contract row by id. If include\_deleted is False, only returns non-deleted rows.  
  
Returns None if not found. The returned tuple follows the `get\_all\_contracts` shape when possible.

Summary:

Return a single contract row by id. If include\_deleted is False, only returns non-deleted rows.

### soft\_delete\_contract

def soft\_delete\_contract(contract\_id, cascade)

Docstring:

Mark a contract (and optionally descendants) as deleted (soft delete).  
  
This sets contracts.deleted = 1 and deleted\_at to now(). Subsets are not removed so they can be restored with the contract.

Summary:

Mark a contract (and optionally descendants) as deleted (soft delete).

Side-effects: Modifies database

### restore\_contract

def restore\_contract(contract\_id, cascade)

Docstring:

Restore a soft-deleted contract (and optionally descendants).  
  
Clears deleted flag and deleted\_at timestamp.

Summary:

Restore a soft-deleted contract (and optionally descendants).

Side-effects: Modifies database

### list\_trashed\_contracts

def list\_trashed\_contracts()

Summary:

No summary available.

### purge\_deleted\_older\_than

def purge\_deleted\_older\_than(days)

Docstring:

Permanently remove soft-deleted contracts older than `days` days.  
  
Returns number of contracts purged.

Summary:

Permanently remove soft-deleted contracts older than `days` days.

### delete\_contract\_and\_descendants

def delete\_contract\_and\_descendants(contract\_id)

Docstring:

Recursively delete a contract, its subsets, subset history, and all descendant contracts.  
  
This operates in a single transaction to ensure consistency.

Summary:

Recursively delete a contract, its subsets, subset history, and all descendant contracts.

Side-effects: Modifies database

### update\_subset\_status

def update\_subset\_status(subset\_id, new\_status, actor\_user\_id)

Docstring:

Update subset status and record history.  
  
actor\_user\_id is required (audit) and must be one of allowed roles.

Summary:

Update subset status and record history.

Side-effects: Modifies database

### get\_subset\_status\_history

def get\_subset\_status\_history(subset\_id)

Summary:

No summary available.

### get\_status\_color

def get\_status\_color(status)

Summary:

No summary available.

### \_hash\_password

def \_hash\_password(password, salt)

Summary:

No summary available.

### get\_admin\_user

def get\_admin\_user()

Docstring:

Return the admin user row, or None if no admin exists.

Summary:

Return the admin user row, or None if no admin exists.

### create\_user

def create\_user(email, password, role)

Summary:

No summary available.

Side-effects: Modifies database

### get\_user\_by\_email

def get\_user\_by\_email(email)

Summary:

No summary available.

### get\_user\_by\_id

def get\_user\_by\_id(user\_id)

Summary:

No summary available.

### verify\_user

def verify\_user(email, password)

Summary:

No summary available.

### create\_reset\_token

def create\_reset\_token(email, hours\_valid)

Summary:

No summary available.

Side-effects: Modifies database

### reset\_password\_with\_token

def reset\_password\_with\_token(token, new\_password)

Summary:

No summary available.

Side-effects: Modifies database

### send\_email

def send\_email(to\_email, subject, body)

Docstring:

Send an email using settings in src/email\_config.py.  
Raises RuntimeError with a helpful message on failure.

Summary:

Send an email using settings in src/email\_config.py.  
Raises RuntimeError with a helpful message on failure.

Side-effects: Sends email (side-effect); Performs file I/O or opens files

### send\_password\_reset\_email

def send\_password\_reset\_email(email, token)

Summary:

No summary available.

Side-effects: Sends email (side-effect)

### generate\_verification\_code

def generate\_verification\_code()

Summary:

No summary available.

### send\_verification\_code

def send\_verification\_code(email, code)

Summary:

No summary available.

Side-effects: Sends email (side-effect)

### enqueue\_email\_outbox

def enqueue\_email\_outbox(to\_email, subject, body, raw\_message, tracking\_code, mark\_failed, last\_error)

Docstring:

Insert an email into the DB outbox and return the outbox row id.

Summary:

Insert an email into the DB outbox and return the outbox row id.

Side-effects: Modifies database

### mark\_outbox\_sent

def mark\_outbox\_sent(outbox\_id)

Summary:

No summary available.

Side-effects: Modifies database

### mark\_outbox\_failed

def mark\_outbox\_failed(outbox\_id, last\_error)

Summary:

No summary available.

Side-effects: Modifies database

### mark\_outbox\_sent\_by\_raw

def mark\_outbox\_sent\_by\_raw(raw\_message)

Summary:

No summary available.

### process\_outbox\_once

def process\_outbox\_once(max\_attempts, backoff\_base)

Docstring:

Process pending outbox rows once. Returns number of processed rows.  
  
This function will attempt to send pending emails (status='pending') and update their status.  
It will skip rows that have attempt\_count >= max\_attempts.

Summary:

Process pending outbox rows once. Returns number of processed rows.

Side-effects: Sends email (side-effect)

### start\_outbox\_worker

def start\_outbox\_worker(poll\_interval)

Docstring:

Start a background thread that processes the outbox periodically.  
  
This is safe to call multiple times; it will start only one worker thread.

Summary:

Start a background thread that processes the outbox periodically.

### stop\_outbox\_worker

def stop\_outbox\_worker()

Docstring:

Stop the background outbox worker thread if running.

Summary:

Stop the background outbox worker thread if running.

### create\_employee

def create\_employee(user\_id, name, dob, job\_title, role, year\_start, profile\_pic, contract\_type, year\_end)

Docstring:

Creates an employee row. Accepts keyword args (used by GUI/signup).

Summary:

Creates an employee row. Accepts keyword args (used by GUI/signup).

Side-effects: Modifies database

### get\_employee\_by\_user

def get\_employee\_by\_user(user\_id)

Summary:

No summary available.

### get\_employee\_by\_id

def get\_employee\_by\_id(emp\_id)

Summary:

No summary available.

### search\_employees

def search\_employees(term, limit)

Docstring:

Search employees by name, job\_title, role, or employee\_number.  
  
term: free-text; numeric strings will match employee\_number and id.  
Returns list of rows: id, user\_id, employee\_number, name, dob, job\_title, role, year\_start, year\_end, profile\_pic, contract\_type

Summary:

Search employees by name, job\_title, role, or employee\_number.

### update\_employee

def update\_employee(emp\_id, \*\*kwargs)

Summary:

No summary available.

Side-effects: Modifies database

### has\_checkin\_today

def has\_checkin\_today(employee\_id)

Summary:

No summary available.

### has\_open\_session

def has\_open\_session(employee\_id)

Summary:

No summary available.

### has\_checked\_out\_today

def has\_checked\_out\_today(employee\_id)

Docstring:

Return True if the employee has a non-null check\_out timestamp recorded during today's UTC date.

Summary:

Return True if the employee has a non-null check\_out timestamp recorded during today's UTC date.

### record\_check\_in

def record\_check\_in(employee\_id)

Summary:

No summary available.

Side-effects: Modifies database

### record\_check\_out

def record\_check\_out(employee\_id)

Summary:

No summary available.

Side-effects: Modifies database

### get\_work\_seconds\_in\_period

def get\_work\_seconds\_in\_period(employee\_id, start\_iso, end\_iso)

Summary:

No summary available.

### get\_month\_work\_seconds

def get\_month\_work\_seconds(employee\_id, year, month)

Summary:

No summary available.

### calculate\_salary

def calculate\_salary(employee\_id, start\_date, end\_date, hourly\_wage)

Summary:

No summary available.

### submit\_pending\_contract

def submit\_pending\_contract(contract\_id, employee\_id, construction\_id, parent\_contract\_id, area, incharge, start\_date, end\_date, terms, file\_path, submitted\_by)

Summary:

No summary available.

Side-effects: Modifies database

### list\_pending\_contracts

def list\_pending\_contracts(status)

Summary:

No summary available.

### \_promote\_pending\_to\_contract

def \_promote\_pending\_to\_contract(row)

Summary:

No summary available.

### approve\_pending\_contract

def approve\_pending\_contract(pending\_id, approved\_by)

Summary:

No summary available.

Side-effects: Modifies database

### reject\_pending\_contract

def reject\_pending\_contract(pending\_id, rejected\_by, reason)

Summary:

No summary available.

Side-effects: Modifies database

### get\_all\_users

def get\_all\_users()

Summary:

No summary available.

### update\_user\_role

def update\_user\_role(user\_id, new\_role, actor\_user\_id)

Docstring:

Update a user's role. Optionally record who performed the change (actor\_user\_id) in the role\_audit table.

Summary:

Update a user's role. Optionally record who performed the change (actor\_user\_id) in the role\_audit table.

Side-effects: Modifies database

### delete\_user\_with\_admin\_check

def delete\_user\_with\_admin\_check(user\_id, transfer\_to\_user\_id)

Docstring:

Prevents deletion of an admin account unless the role is transferred to another user.  
Returns True if deletion succeeded, False otherwise.

Summary:

Prevents deletion of an admin account unless the role is transferred to another user.  
Returns True if deletion succeeded, False otherwise.

Side-effects: Modifies database

### delete\_user

def delete\_user(user\_id)

Summary:

No summary available.

Side-effects: Modifies database

### can\_edit

def can\_edit(target\_role, actor\_role)

Docstring:

Can actor edit target? Only if actor's role is higher.

Summary:

Can actor edit target? Only if actor's role is higher.

### can\_delete

def can\_delete(target\_role, actor\_role)

Docstring:

Only admin can delete users (other than admin)

Summary:

Only admin can delete users (other than admin)

### can\_view\_salary

def can\_view\_salary(actor\_role)

Docstring:

Who can view salary? Admin, high\_manager, accountant.

Summary:

Who can view salary? Admin, high\_manager, accountant.

### can\_count\_salary

def can\_count\_salary(actor\_role)

Docstring:

Who can use salary counting? Only accountant.

Summary:

Who can use salary counting? Only accountant.

### can\_grant\_role

def can\_grant\_role(actor\_role, target\_role)

Docstring:

Can actor grant target\_role? Actor must have higher hierarchy value.  
Non-admins cannot grant admin or high\_manager roles.

Summary:

Can actor grant target\_role? Actor must have higher hierarchy value.  
Non-admins cannot grant admin or high\_manager roles.

### can\_edit\_info

def can\_edit\_info(actor\_role, target\_role)

Summary:

No summary available.

### can\_view\_working\_hours

def can\_view\_working\_hours(actor\_role, target\_user\_id, actor\_user\_id)

Docstring:

Accountant can view all, engineer only self, managers view below their level.

Summary:

Accountant can view all, engineer only self, managers view below their level.

## hr\_management\_app\src\email\_config.py

SMTP configuration.  
  
This file reads settings from environment variables to avoid committing secrets into  
source control. To run locally, set these env vars (for example in PowerShell):  
  
# set SMTP env vars (example)  
# $env:SMTP\_SERVER = 'smtp.gmail.com'  
# $env:SMTP\_PORT = '465'  
# $env:SMTP\_USE\_SSL = 'True'  
# $env:SMTP\_USER = 'you@example.com'  
# $env:SMTP\_PASSWORD = 'your-app-password'  
# $env:FROM\_EMAIL = 'you@example.com'  
  
If you prefer a local file, create `email\_config\_local.py` (gitignored) with the same  
names and values and import it from here instead. By default values are read from the  
environment and fall back to safe defaults.

## hr\_management\_app\src\employees\\_\_init\_\_.py

No file-level docstring available.

## hr\_management\_app\src\employees\models.py

No file-level docstring available.

### Employee

class Employee

Summary:

No summary available.

## hr\_management\_app\src\gui.py

No file-level docstring available.

### EmployeeProfileWindow

class EmployeeProfileWindow(tk.Toplevel)

Summary:

No summary available.

### EmployeeManagementWindow

class EmployeeManagementWindow(tk.Toplevel)

Summary:

No summary available.

### ManageUsersWindow

class ManageUsersWindow(tk.Toplevel)

Summary:

No summary available.

### HRApp

class HRApp(tk.Tk)

Summary:

No summary available.

## hr\_management\_app\src\main.py

No file-level docstring available.

### check\_in

def check\_in(employee\_id)

Summary:

No summary available.

### check\_out

def check\_out(employee\_id)

Summary:

No summary available.

### add\_contract

def add\_contract()

Summary:

No summary available.

### load\_contracts\_from\_db

def load\_contracts\_from\_db()

Summary:

No summary available.

### attendance\_menu

def attendance\_menu()

Summary:

No summary available.

### salary\_menu

def salary\_menu()

Summary:

No summary available.

### cli\_main

def cli\_main()

Summary:

No summary available.

## hr\_management\_app\src\ml\\_\_init\_\_.py

ML package shim to help tools and static analysis resolve imports like `ml.imputer\_ml`.  
This keeps imports lazy and avoids importing heavy ML dependencies at module import time.

### \_\_getattr\_\_

def \_\_getattr\_\_(name)

Summary:

No summary available.

## hr\_management\_app\src\ml\imputer.py

Simple missing-data imputation utilities used by the import pipeline.  
  
Strategy:  
- Build light statistics from existing DB and/or the current batch.  
- For categorical fields (job\_title, role, contract\_type): use most common value.  
- For numeric fields (year\_start): use median or mode depending on distribution.  
- For email: if missing but name exists, synthesize using a safe pattern (name + unique suffix) when allowed; otherwise leave None.  
- For dob: attempt to infer year from year\_start - typical career start age (e.g., 22-30) or leave None.  
  
This module is intentionally small and deterministic so it works without heavy ML libs.

### most\_common

def most\_common(values, default)

Summary:

No summary available.

### median\_int

def median\_int(values, default)

Summary:

No summary available.

### synthesize\_email\_from\_name

def synthesize\_email\_from\_name(name, existing\_emails)

Docstring:

Create a safe synthesized email using name and a numeric suffix if needed.

Summary:

Create a safe synthesized email using name and a numeric suffix if needed.

### infer\_missing\_fields

def infer\_missing\_fields(batch, db\_stats)

Docstring:

Take a batch of cleaned records and return a new list with imputed values.  
  
db\_stats: optional precomputed stats such as {"job\_title\_common": ..., "year\_start\_median": ..., "emails": [...]}

Summary:

Take a batch of cleaned records and return a new list with imputed values.

## hr\_management\_app\src\ml\imputer\_heuristic.py

Heuristic imputer for missing fields.  
Provides:  
- fit\_from\_records(records): build frequency maps and medians  
- predict\_batch(records, model): fill missing fields using heuristics  
  
Heuristics:  
- job\_title: most common job\_title for the same role, else global most common  
- year\_start: median year for the job\_title, else global median, else 2008  
- email: if missing and name present -> name-based email with domain example.com  
- name: if missing and email present -> infer from local-part

### \_normalize\_role

def \_normalize\_role(r)

Summary:

No summary available.

### fit\_from\_records

def fit\_from\_records(records)

Summary:

No summary available.

### \_email\_from\_name

def \_email\_from\_name(name, idx)

Summary:

No summary available.

### \_name\_from\_email

def \_name\_from\_email(email)

Summary:

No summary available.

### predict\_batch

def predict\_batch(records, model)

Summary:

No summary available.

## hr\_management\_app\src\ml\imputer\_ml.py

ML-backed imputer for missing job\_title and year\_start.  
  
This file provides:  
- fit\_imputer\_from\_db(conn\_or\_records): trains simple sklearn models (RandomForest)  
 to predict job\_title (categorical) and year\_start (regression) from available fields.  
- load\_model(path): loads saved models (job and year models) using joblib.  
- predict\_batch(records, model): predicts missing fields for a batch and returns filled records + confidences.  
  
Design choices:  
- Feature extraction is intentionally simple: name token counts, role one-hot (mapped), and simple length features.  
- Categorical job\_title prediction uses a RandomForestClassifier with label encoding.  
- Year\_start prediction uses RandomForestRegressor.  
- Model artifacts are saved together in a directory with joblib.  
  
Note: This is a lightweight approach suited for medium-sized datasets (thousands of rows). For 20k+ rows  
it will work better; tune hyperparameters if needed.

### \_name\_tokens

def \_name\_tokens(name)

Summary:

No summary available.

### \_extract\_features

def \_extract\_features(records, le\_role)

Summary:

No summary available.

### fit\_imputer\_from\_records

def fit\_imputer\_from\_records(records, save\_to)

Docstring:

Train models from provided records. Returns a dict with models and label encoders.

Summary:

Train models from provided records. Returns a dict with models and label encoders.

Side-effects: Performs file I/O or opens files

### load\_model

def load\_model(path)

Summary:

No summary available.

Side-effects: Performs file I/O or opens files

### predict\_batch

def predict\_batch(records, model)

Summary:

No summary available.

## hr\_management\_app\src\ml\ner.py

Lightweight NER wrapper used by the import pipeline.  
  
This module attempts to load spaCy and a language model. If spaCy is not  
available, it provides a conservative rule-based fallback so the rest of the  
app can function without ML dependencies.

### extract\_entities\_spacy

def extract\_entities\_spacy(text)

Docstring:

Extract a best-effort mapping from free text using spaCy NER.  
  
Returns a dict with possible keys: name, email, dob, job\_title, role, year\_start, year\_end, contract\_type

Summary:

Extract a best-effort mapping from free text using spaCy NER.

### extract\_entities\_fallback

def extract\_entities\_fallback(text)

Docstring:

Conservative rule-based fallback for extracting a few fields from text.

Summary:

Conservative rule-based fallback for extracting a few fields from text.

### extract\_entities

def extract\_entities(text)

Docstring:

Public API: try spaCy, else fallback.

Summary:

Public API: try spaCy, else fallback.

## hr\_management\_app\src\ml\train\_spacy\_ner.py

Helper script to train a spaCy NER model for extracting employee fields.  
  
Usage (local dev):  
 - Prepare training data in spaCy JSONL format with entities for labels: NAME, EMAIL, DOB, JOB\_TITLE, ROLE, YEAR\_START, YEAR\_END, CONTRACT\_TYPE  
 - Install spaCy and a base model: `pip install spacy && python -m spacy download en\_core\_web\_sm`  
 - Run: `python ml/train\_spacy\_ner.py --train data/train.spacy --output ./models/employee\_ner --n\_iter 20`  
  
This script is intentionally lightweight; it checks for spaCy and exits with a helpful message if not available.

### main

def main()

Summary:

No summary available.

Side-effects: Performs file I/O or opens files

## hr\_management\_app\src\parsers\\_\_init\_\_.py

Parsers package shim to help tools and static analysis resolve imports.  
This file intentionally keeps imports lazy to avoid heavy dependencies at import time.

### \_\_getattr\_\_

def \_\_getattr\_\_(name)

Summary:

No summary available.

## hr\_management\_app\src\parsers\file\_parser.py

No file-level docstring available.

### parse\_excel

def parse\_excel(path)

Docstring:

Read an Excel file and return list of row dicts (columns as-is).

Summary:

Read an Excel file and return list of row dicts (columns as-is).

### parse\_csv

def parse\_csv(path)

Summary:

No summary available.

### parse\_docx

def parse\_docx(path)

Docstring:

Very small docx parser: extract any tables and return rows; fallback to paragraphs.  
Returns list of dicts where keys are column headers when a table is found.

Summary:

Very small docx parser: extract any tables and return rows; fallback to paragraphs.  
Returns list of dicts where keys are column headers when a table is found.

## hr\_management\_app\src\parsers\mapping\_store.py

No file-level docstring available.

### \_default\_path

def \_default\_path()

Summary:

No summary available.

### load\_config

def load\_config(path)

Summary:

No summary available.

Side-effects: Performs file I/O or opens files

### save\_config

def save\_config(config, path)

Summary:

No summary available.

Side-effects: Performs file I/O or opens files

## hr\_management\_app\src\parsers\normalizer.py

No file-level docstring available.

### \_normalize\_key

def \_normalize\_key(k)

Summary:

No summary available.

### map\_columns

def map\_columns(row, fuzzy\_threshold)

Docstring:

Map input row keys to canonical field names. fuzzy\_threshold controls minimum score for fuzzy matches.

Summary:

Map input row keys to canonical field names. fuzzy\_threshold controls minimum score for fuzzy matches.

### map\_columns\_debug

def map\_columns\_debug(row, fuzzy\_threshold)

Docstring:

Like map\_columns but also return a debug map of original\_key -> (canonical, score).  
Score is an int 0-100 when fuzzy matching was used, or None for exact matches.

Summary:

Like map\_columns but also return a debug map of original\_key -> (canonical, score).  
Score is an int 0-100 when fuzzy matching was used, or None for exact matches.

### validate\_and\_clean

def validate\_and\_clean(record)

Docstring:

Return cleaned record and list of validation problems (empty if ok).

Summary:

Return cleaned record and list of validation problems (empty if ok).

## hr\_management\_app\src\smoke\_test.py

Quick smoke test: instantiate DB helpers and run simple operations without GUI or network.  
  
This script will:  
- import database helpers  
- create a temporary user and employee (using a random email)  
- record a check-in and check-out  
- calculate month work seconds for current month  
- print results  
  
Run inside the project's venv Python as provided by the environment.

## hr\_management\_app\src\test\_import\_integration.py

No file-level docstring available.

### test\_import\_flow\_with\_fuzzy\_headers

def test\_import\_flow\_with\_fuzzy\_headers(tmp\_path)

Summary:

No summary available.

Side-effects: Performs file I/O or opens files

## hr\_management\_app\src\test\_smtp.py

No file-level docstring available.

### test\_smtp

def test\_smtp()

Summary:

No summary available.

Side-effects: Sends email (side-effect)

## hr\_management\_app\src\test\_ui\_import\_dialog.py

No file-level docstring available.

### test\_importdialog\_load\_and\_import

def test\_importdialog\_load\_and\_import(tmp\_path, monkeypatch)

Summary:

No summary available.

Side-effects: Performs file I/O or opens files

## hr\_management\_app\src\tests\test\_contracts.py

No file-level docstring available.

### setup\_module

def setup\_module()

Summary:

No summary available.

### test\_contract\_progress\_and\_permissions

def test\_contract\_progress\_and\_permissions(tmp\_path)

Summary:

No summary available.

## hr\_management\_app\src\tests\test\_contracts\_colors.py

No file-level docstring available.

### test\_status\_colors\_and\_history

def test\_status\_colors\_and\_history(tmp\_path)

Summary:

No summary available.

## hr\_management\_app\src\tests\test\_contracts\_soft\_delete.py

No file-level docstring available.

### setup\_module

def setup\_module()

Summary:

No summary available.

### test\_soft\_delete\_and\_restore\_cycle

def test\_soft\_delete\_and\_restore\_cycle()

Summary:

No summary available.

### teardown\_module

def teardown\_module()

Summary:

No summary available.

## hr\_management\_app\src\tests\test\_database.py

No file-level docstring available.

### test\_create\_user\_and\_employee

def test\_create\_user\_and\_employee()

Summary:

No summary available.

### test\_checkin\_checkout\_and\_month\_seconds

def test\_checkin\_checkout\_and\_month\_seconds()

Summary:

No summary available.

### test\_contract\_requires\_existing\_employee

def test\_contract\_requires\_existing\_employee()

Summary:

No summary available.

### test\_create\_employee\_invalid\_year

def test\_create\_employee\_invalid\_year()

Summary:

No summary available.

## hr\_management\_app\src\tests\test\_ml\_ner.py

No file-level docstring available.

### test\_ner\_fallback\_extracts\_email\_and\_name

def test\_ner\_fallback\_extracts\_email\_and\_name()

Summary:

No summary available.

## hr\_management\_app\src\tests\test\_normalizer.py

No file-level docstring available.

### test\_map\_and\_validate

def test\_map\_and\_validate()

Summary:

No summary available.

## hr\_management\_app\src\tests\test\_parsers.py

No file-level docstring available.

### test\_parse\_csv\_basic

def test\_parse\_csv\_basic()

Summary:

No summary available.

Side-effects: Performs file I/O or opens files

## hr\_management\_app\src\tests\test\_role\_audit\_and\_permissions.py

No file-level docstring available.

### test\_can\_grant\_role\_logic

def test\_can\_grant\_role\_logic()

Summary:

No summary available.

### test\_role\_audit\_records\_actor

def test\_role\_audit\_records\_actor()

Summary:

No summary available.

## hr\_management\_app\src\tests\test\_search\_contracts\_and\_employees.py

No file-level docstring available.

### setup\_test\_db

def setup\_test\_db()

Summary:

No summary available.

### teardown\_test\_db

def teardown\_test\_db(path)

Summary:

No summary available.

### test\_contract\_search\_basic

def test\_contract\_search\_basic()

Summary:

No summary available.

### test\_employee\_search\_basic

def test\_employee\_search\_basic()

Summary:

No summary available.

Side-effects: Modifies database

## hr\_management\_app\src\tests\test\_user\_management.py

No file-level docstring available.

### test\_update\_user\_role\_admin\_conflict

def test\_update\_user\_role\_admin\_conflict()

Summary:

No summary available.

### test\_delete\_admin\_requires\_transfer\_and\_transfers\_role

def test\_delete\_admin\_requires\_transfer\_and\_transfers\_role()

Summary:

No summary available.

## hr\_management\_app\src\tests\test\_validators.py

No file-level docstring available.

### test\_validate\_contract\_fields\_ok

def test\_validate\_contract\_fields\_ok()

Summary:

No summary available.

### test\_validate\_contract\_fields\_bad

def test\_validate\_contract\_fields\_bad(cid, eid, start, end)

Summary:

No summary available.

## hr\_management\_app\src\ui\_helpers.py

No file-level docstring available.

### role\_selection\_dialog

def role\_selection\_dialog(parent, email, current\_role, allowed)

Docstring:

Modal dialog to pick a role from allowed list. Returns selected role or None.

Summary:

Modal dialog to pick a role from allowed list. Returns selected role or None.

## hr\_management\_app\src\ui\_import.py

Minimal import UI for bulk employee import.  
  
Provides a compact, syntactically-correct implementation of the import  
dialogs used by the main GUI. This is intentionally small and easy to  
unit-test.

### \_collect\_db\_stats

def \_collect\_db\_stats()

Docstring:

Collect lightweight stats from users/employees to help imputation.

Summary:

Collect lightweight stats from users/employees to help imputation.

### \_safe\_int

def \_safe\_int(val)

Summary:

No summary available.

### ImportDialog

class ImportDialog(tk.Toplevel)

Summary:

No summary available.

### MappingPreviewDialog

class MappingPreviewDialog(tk.Toplevel)

Summary:

No summary available.

### SettingsDialog

class SettingsDialog(tk.Toplevel)

Summary:

No summary available.

### EditRowDialog

class EditRowDialog(tk.Toplevel)

Summary:

No summary available.

### ImputationPreviewDialog

class ImputationPreviewDialog(tk.Toplevel)

Docstring:

Modal dialog to preview proposed imputations and accept/reject them (with inline editing).

Summary:

Modal dialog to preview proposed imputations and accept/reject them (with inline editing).

## hr\_management\_app\src\ui\_validators.py

No file-level docstring available.

### \_contract\_exists

def \_contract\_exists(contract\_id)

Summary:

No summary available.

### \_detect\_cycle

def \_detect\_cycle(start\_id, parent\_id)

Docstring:

Detect whether assigning parent\_id as parent of start\_id would create a cycle.  
  
Walk up the parent chain from parent\_id and see if we encounter start\_id.

Summary:

Detect whether assigning parent\_id as parent of start\_id would create a cycle.

### validate\_contract\_fields

def validate\_contract\_fields(cid\_text, eid\_text, start, end, parent\_text)

Docstring:

Validate and parse contract input strings.  
  
Returns (cid, eid, start\_iso, end\_iso, parent\_id) or raises ValueError.  
parent\_text may be None or empty; when provided, parent existence and cycles are checked.

Summary:

Validate and parse contract input strings.

## hr\_management\_app\tests\test\_contract\_trash.py

No file-level docstring available.

### ContractTrashTests

class ContractTrashTests(unittest.TestCase)

Summary:

No summary available.

## hr\_management\_app\tests\test\_imputation\_audit.py

No file-level docstring available.

### ImputationAuditTests

class ImputationAuditTests(unittest.TestCase)

Summary:

No summary available.

## hr\_management\_app\tests\test\_integration\_import\_flow.py

No file-level docstring available.

### IntegrationImportFlowTests

class IntegrationImportFlowTests(unittest.TestCase)

Summary:

No summary available.

## hr\_management\_app\tests\test\_ml\_imputer.py

No file-level docstring available.

### MLImputerTests

class MLImputerTests(unittest.TestCase)

Summary:

No summary available.

## hr\_management\_app\tests\test\_parsers\_csv.py

No file-level docstring available.

### ParserCsvTests

class ParserCsvTests(unittest.TestCase)

Summary:

No summary available.

## hr\_management\_app\tests\test\_pending\_contracts\_gui\_integration.py

No file-level docstring available.

### \_setup\_temp\_db

def \_setup\_temp\_db(tmp\_path)

Summary:

No summary available.

### \_reload\_gui\_module

def \_reload\_gui\_module()

Summary:

No summary available.

### test\_pending\_contract\_window\_selects\_focused\_id

def test\_pending\_contract\_window\_selects\_focused\_id(tmp\_path)

Summary:

No summary available.

## hr\_management\_app\tests\test\_pending\_contracts\_pytest.py

No file-level docstring available.

### \_load\_database\_with\_temp\_db

def \_load\_database\_with\_temp\_db(tmp\_path)

Summary:

No summary available.

### test\_submit\_and\_approve\_pending\_contract\_non\_gui

def test\_submit\_and\_approve\_pending\_contract\_non\_gui(tmp\_path)

Summary:

No summary available.

### test\_pending\_selection\_indexing

def test\_pending\_selection\_indexing(tmp\_path)

Summary:

No summary available.

## hr\_management\_app\tmp\_check\_test.py

No file-level docstring available.

## hr\_management\_app\tmp\_gui\_smoke\_sim.py

Non-visual GUI smoke simulation: exercises check-in/check-out flow using database helpers  
and prints the messages the GUI would display plus attendance DB rows for inspection.

### dump\_attendance

def dump\_attendance(eid)

Summary:

No summary available.

## hr\_management\_app\tmp\_pending\_contract\_smoke.py

Smoke test: submit pending contract as low-role user, approve it as manager, assert live contract exists.  
Run with PYTHONPATH set to the repository root. For example, set PYTHONPATH to your repo root and run this script.

## hr\_management\_app\tools\apply\_heuristic\_imputer.py

Apply heuristic imputer to an XLSX file and produce an imputed file + report.  
Usage: use the project venv python to run this script.

### read\_xlsx

def read\_xlsx(path)

Summary:

No summary available.

### write\_xlsx

def write\_xlsx(records, path)

Summary:

No summary available.

## hr\_management\_app\tools\check\_sklearn\_import.py

Diagnostic: check sklearn import and imputer\_ml HAS\_SKLEARN flag.

## hr\_management\_app\tools\force\_train\_sklearn.py

Force a sklearn-based training run and save joblib artifact.  
Run from repo root: python hr\_management\_app/tools/force\_train\_sklearn.py

### fetch\_records

def fetch\_records(limit)

Summary:

No summary available.

## hr\_management\_app\tools\generate\_dummy\_20k.py

Generate a dummy Excel file with 20,000 people for ML testing.  
Creates: hr\_management\_app/data/dummy\_import\_20k.xlsx  
Columns: email,name,job\_title,role,year\_start  
Approximately 1,000 rows will have 1-3 random missing fields.  
  
Run with the project venv: ./.venv/Scripts/python.exe hr\_management\_app/tools/generate\_dummy\_20k.py

### random\_name

def random\_name()

Summary:

No summary available.

### random\_email

def random\_email(name, idx)

Summary:

No summary available.

### random\_year\_start

def random\_year\_start()

Summary:

No summary available.

### create\_workbook

def create\_workbook()

Summary:

No summary available.

## hr\_management\_app\tools\generate\_dummy\_xlsx.py

Generate a dummy XLSX with 2000 people for ML testing.  
  
Creates: hr\_management\_app/src/tests/fixtures/dummy\_employees\_2000.xlsx  
Columns: NAME, EMAIL, DOB, JOB\_TITLE, ROLE, YEAR\_START, YEAR\_END, CONTRACT\_TYPE

### random\_dob

def random\_dob()

Summary:

No summary available.

### random\_year\_start

def random\_year\_start()

Summary:

No summary available.

### random\_year\_end

def random\_year\_end(start)

Summary:

No summary available.

## hr\_management\_app\tools\launch\_import\_gui.py

Launch the ImportDialog UI with the generated XLSX preloaded.  
  
This will open a Tkinter window. Use this when running on a desktop environment.

## hr\_management\_app\tools\preview\_imputations\_headless.py

Headless preview of imputation proposals for debugging the UI preview flow.  
Prints counts of proposed fields and sample proposals.

### read\_xlsx\_simple

def read\_xlsx\_simple(path)

Summary:

No summary available.

## hr\_management\_app\tools\run\_import\_xlsx.py

Headless importer: load an XLSX file and run the same pipeline used by the UI import.  
  
Usage: adjust INPUT\_PATH constant or pass via environment/PYTHONPATH and run.

## hr\_management\_app\tools\run\_smoke\_impute.py

No file-level docstring available.

## hr\_management\_app\tools\smoke\_imports.py

Simple smoke-test: import core modules and report import-time errors.  
Run from project root with the virtualenv Python so imports resolve.

## hr\_management\_app\tools\smoke\_impute.py

Smoke test: parse the existing dummy XLSX and show imputation results for first N rows.

## hr\_management\_app\tools\smoke\_test\_imputer.py

Smoke test for imputer model: load model and run predict\_batch on sample records.

### main

def main()

Summary:

No summary available.

## hr\_management\_app\tools\train\_imputer.py

Train the ML imputer model from DB records and save artifact.  
  
Usage: run from repo root with virtualenv activated:  
  
 python hr\_management\_app/tools/train\_imputer.py  
  
This script will read employees (and users) from the DB, build a training set,  
train models and save them under src/models/imputer\_model.joblib

### fetch\_records

def fetch\_records(limit)

Summary:

No summary available.

## hr\_management\_app\tools\train\_imputer\_from\_xlsx.py

Train imputer models from an XLSX file and save sklearn joblib artifact.  
  
Usage: run with the project's python executable, e.g.:  
 .venv/Scripts/python.exe hr\_management\_app/tools/train\_imputer\_from\_xlsx.py

### read\_xlsx

def read\_xlsx(path)

Summary:

No summary available.

## Test Summary

................................... [100%]  
============================== warnings summary ===============================  
..\..\..\AppData\Roaming\Python\Python313\site-packages\spacy\cli\\_util.py:23  
 C:\Users\DELL\AppData\Roaming\Python\Python313\site-packages\spacy\cli\\_util.py:23: DeprecationWarning: Importing 'parser.split\_arg\_string' is deprecated, it will only be available in 'shell\_completion' in Click 9.0.  
 from click.parser import split\_arg\_string  
  
..\..\..\AppData\Roaming\Python\Python313\site-packages\weasel\util\config.py:8  
 C:\Users\DELL\AppData\Roaming\Python\Python313\site-packages\weasel\util\config.py:8: DeprecationWarning: Importing 'parser.split\_arg\_string' is deprecated, it will only be available in 'shell\_completion' in Click 9.0.  
 from click.parser import split\_arg\_string  
  
hr\_management\_app/tests/test\_contract\_trash.py::ContractTrashTests::test\_delete\_contract\_and\_descendants  
hr\_management\_app/tests/test\_contract\_trash.py::ContractTrashTests::test\_purge\_deleted\_older\_than  
hr\_management\_app/tests/test\_contract\_trash.py::ContractTrashTests::test\_soft\_delete\_and\_restore  
hr\_management\_app/tests/test\_imputation\_audit.py::ImputationAuditTests::test\_export\_contains\_test\_row  
hr\_management\_app/tests/test\_integration\_import\_flow.py::IntegrationImportFlowTests::test\_end\_to\_end\_import\_flow  
hr\_management\_app/tests/test\_ml\_imputer.py::MLImputerTests::test\_fit\_and\_predict\_fallback  
hr\_management\_app/tests/test\_parsers\_csv.py::ParserCsvTests::test\_parse\_csv\_basic  
 C:\Python313\Lib\unittest\case.py:597: RuntimeWarning: TestResult has no addDuration method  
 warnings.warn("TestResult has no addDuration method",  
  
hr\_management\_app/tests/test\_pending\_contracts\_gui\_integration.py::test\_pending\_contract\_window\_selects\_focused\_id  
hr\_management\_app/tests/test\_pending\_contracts\_pytest.py::test\_submit\_and\_approve\_pending\_contract\_non\_gui  
hr\_management\_app/tests/test\_pending\_contracts\_pytest.py::test\_pending\_selection\_indexing  
hr\_management\_app/tests/test\_pending\_contracts\_pytest.py::test\_pending\_selection\_indexing  
hr\_management\_app/tests/test\_pending\_contracts\_pytest.py::test\_pending\_selection\_indexing  
 C:\Users\DELL\Documents\GitHub\HR\_Management\hr\_management\_app\src\database\database.py:1588: DeprecationWarning: datetime.datetime.utcnow() is deprecated and scheduled for removal in a future version. Use timezone-aware objects to represent datetimes in UTC: datetime.datetime.now(datetime.UTC).  
 submitted\_at = datetime.utcnow().isoformat()  
  
hr\_management\_app/tests/test\_pending\_contracts\_pytest.py::test\_submit\_and\_approve\_pending\_contract\_non\_gui  
 C:\Users\DELL\Documents\GitHub\HR\_Management\hr\_management\_app\src\database\database.py:1643: DeprecationWarning: datetime.datetime.utcnow() is deprecated and scheduled for removal in a future version. Use timezone-aware objects to represent datetimes in UTC: datetime.datetime.now(datetime.UTC).  
 approved\_at = datetime.utcnow().isoformat()  
  
-- Docs: https://docs.pytest.org/en/stable/warnings.html  
35 passed, 15 warnings in 35.03s