

Prudentia Documentation

John Scanlon
Johnscanlon104@gmail.com
087 7662702

Table of Contents

| | |
|------------------------------------|----|
| Introduction | 3 |
| Frontend Features | 4 |
| Frontend Codebase Breakdown | 12 |
| Backend Features | 14 |
| Backend Codebase Breakdown | 15 |
| Database | 16 |
| Development Diary | 18 |
| Website Re-Deployment Instructions | 20 |

Introduction

The link to the Prudentia **website** can be found here:

<https://prudentia.onrender.com>

The link to the public **github** for the Prudentia website can be found here:

<https://github.com/djonskanlyn/Prudentia/>

The **presentation** slides for the Prudentia project can be found here:

https://github.com/djonskanlyn/Prudentia/blob/main/prudentia_docs/prudentia_presentation.pdf

In summary, Prudentia is a tool for financial regulatory supervisors to review quarterly prudential return data submitted by regulated firms. The website allows users to view return data, create PR reviews and 4-eyes sign-off on these reviews.

The Prudentia website has 3 components:

1. Frontend (React via Vite: Javascript)
2. Backend API (Django with REST framework: Python)
3. Database (PostgreSQL)

All three components are hosted on render.

1. The frontend is hosted as a static site.
2. The backend is hosted as a web service.
3. The database is hosted as a PostgreSQL database.

Swagger and Redocly was incorporated into the project to improve interaction and documentation of the API.

You must be logged into the backend as a superuser to interact directly with the API.

The link to the Prudentia backend's **Django Admin** section can be found here:

<https://prudentiaapi.onrender.com/admin/>

(see supplied password list file in UCDDA submission for a superuser username and password).

The link to the **Swagger** implementation for the API, accessible as a logged on superuser, is here: <https://prudentiaapi.onrender.com/swagger/>

The link to the **Redocly** automated API documentation for the project, accessible as a logged on superuser is here: <https://prudentiaapi.onrender.com/redoc/>

Frontend Features

You must logon to the website to be able to view any information. The landing page is the login page. This is because no part of this website is for non-authorized users.

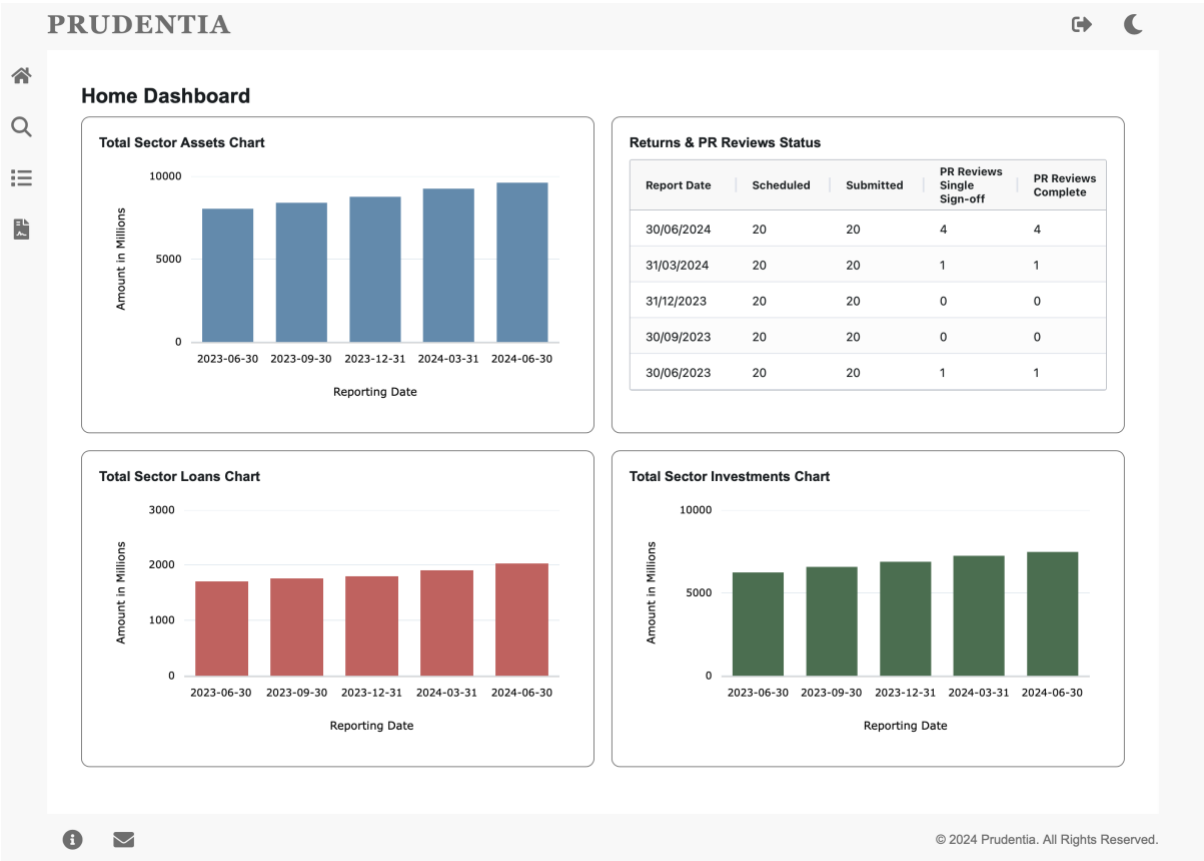
Additionally, this webapp is designed for desktop use only, it is not designed for use on a mobile phone or tablet as this would not be appropriate and was a design decision.

Login Page:

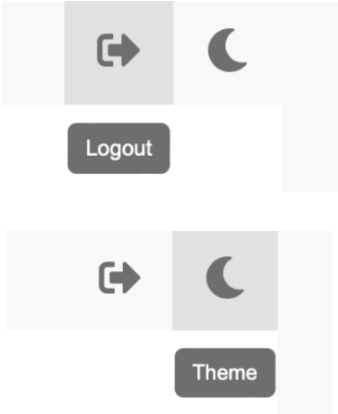
The **Login** page is the landing page, it also provides links to register as a user to the website (note in reality, registration would not be open to anyone, but project specification dictates that a registration link must be provided). Additionally there is a password reset link that takes you to the Django backend to process the password reset via Google API. Completion of reset provides a link to take you back to the frontend. You cannot access any area of the website if you have not signed in, all icons/links will redirect to the login landing page.

Home Dashboard Page:

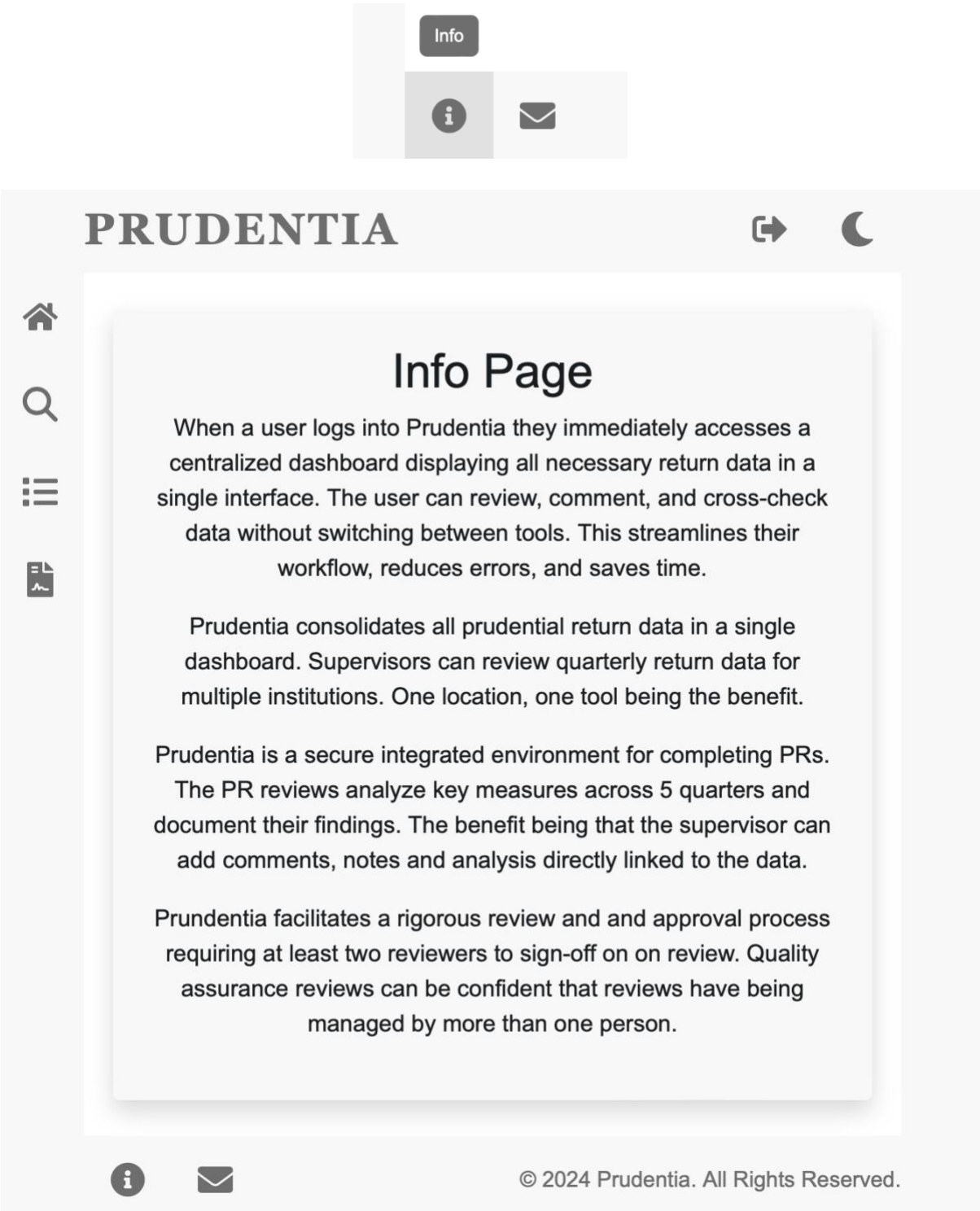
Having logged on the user will be redirected to **Home Dashboard** page. They will be able to navigate to all the sections of the website using the top, bottom, left and right navbars. The dashboard also displays summary graphs of sectoral total assets, sectoral total investments, sectoral total loans and a grid table that summarises by reporting period the number of returns scheduled, submitted, the number of PR reviews with single sign-off, and the number with double sign-off (completed).

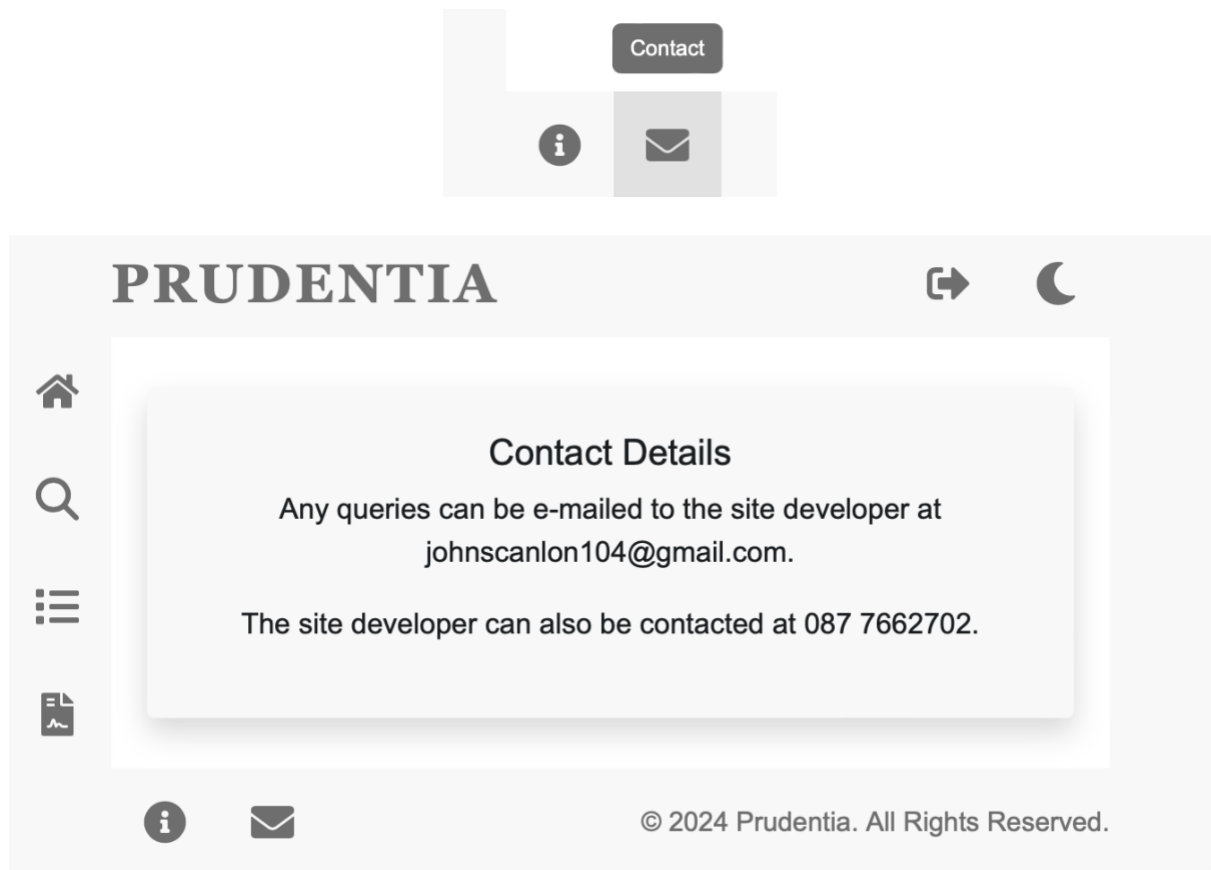


In the top navbar there is a **logout** icon and a website **theme** switch (light/dark) icon.



In the bottom navbar there is an icon that will take the user to an **info** page, and another icon that will take the user to a **contact** page should they need to contact the developer.





In the left navbar there are 4 icons. One will take to user to the **Returns Detail** page, the second will take the user to the **Returns List** page and the third to the **PR Reviews List** page. The fourth will take the user back to the **Home Dashboard** page.



Returns Detail Page:





This page shows a formatted version of the return data submitted by the regulated firms. The user selects a firm and a reporting date, and the data will load.

There are 5 tabs that display the 5 pages of data submitted with each return:

1. Income & Expenditure
2. Balance Sheet
3. Deposits & Investments
4. Credit Risk Disclosures
5. Loan Analysis

Note: You need to select a firm and a reporting date to trigger a data fetch when you first logon to the website even if a default selection is provided.

PRUDENTIA



DEF Institution

2023-12-31

Income & Expenditure



Balance Sheet

Deposits & Investments

Credit Risk Disclosures

Loans Analysis

| INCOME & EXPENDITURE | euro | euro | euro |
|----------------------------|-----------|-----------|-----------|
| Income | | | |
| Member loan interest | 3,931,222 | | |
| Other interest income | 4,818,976 | | |
| Other income | 120,693 | | |
| Total income | | 8,870,891 | |
| Total expenditure | | | |
| Employment costs | 2,380,818 | | |
| Other management costs | 4,125,787 | | |
| Depreciation costs | 293,456 | | |
| Net member loan impairment | 12,363 | | |
| Total expenditure | | 6,812,424 | |
| Surplus or deficit | | | 2,058,467 |



© 2024 Prudentia. All Rights Reserved.

Returns List Page:

This page lists all the ‘current’ status returns available in the database. The primary purpose of this page is to identify the status of a return (submitted, version ref. etc) and to trigger the creation of a PR Review if desired. The grid table makes use of AG Grid which has advanced filtering and allows the reorganisation of the order of the fields displayed.

PRUDENTIA

Returns List

| Id | Firm | Report Date | Status | State | Version | Reviews |
|----|-----------------|-------------|-----------|---------|---------|---------------|
| | | 13/10/20 | | | | |
| 1 | ABC Institution | 30/06/2023 | Submitted | Current | 1 | Create Review |
| 2 | BCD Institution | 30/06/2023 | Submitted | Current | 1 | Create Review |
| 3 | CDE Institution | 30/06/2023 | Submitted | Current | 1 | Create Review |
| 4 | DEF Institution | 30/06/2023 | Submitted | Current | 1 | Create Review |
| 5 | EFG Institution | 30/06/2023 | Submitted | Current | 1 | Create Review |
| 6 | FGH Institution | 30/06/2023 | Submitted | Current | 1 | Create Review |
| 7 | GHI Institution | 30/06/2023 | Submitted | Current | 1 | Create Review |
| 8 | HIJ Institution | 30/06/2023 | Submitted | Current | 1 | Create Review |
| 9 | IJK Institution | 30/06/2023 | Submitted | Current | 1 | Create Review |
| 10 | JKL Institution | 30/06/2023 | Submitted | Current | 1 | Create Review |
| 11 | KLM Institution | 30/06/2023 | Submitted | Current | 1 | Create Review |
| 12 | LMN Institution | 30/06/2023 | Submitted | Current | 1 | Create Review |
| 13 | MNO Institution | 30/06/2023 | Submitted | Current | 1 | Create Review |
| 14 | NOP Institution | 30/06/2023 | Submitted | Current | 1 | Create Review |

Page Size: 1001 to 100 of 100Page 1 of 1

© 2024 Prudentia. All Rights Reserved.

| Id | Firm | Report Date | Status | State | Version | Reviews |
|----|-----------------|-------------|-----------|---------|---------|---------------|
| | | 13/10/20 | | | | |
| 1 | ABC Institution | 30/06/2023 | Submitted | Current | 1 | Create Review |
| 2 | BCD Institution | 30/06/2023 | Submitted | Current | 1 | Create Review |

PR Reviews List Page:

This page lists the PR Reviews that have been created and allows all users to view the PR Review Details. The AG Grid based list has a field with a button that lets users view the PR Review by clicking on it.

PRUDENTIA

PR Reviews List

| Review | Return | Firm | Reporting Date | State | Version | Created At | Updated At | Actions |
|--------|--------|-----------------|----------------|---------|---------|---------------------|---------------------|-------------------------------|
| | | | 13/10/2024 | | | | | |
| 6 | 64 | DEF Institution | 31/03/2024 | Current | 1 | 2024-10-11 12:25:59 | 2024-10-11 12:32:53 | <button>View Details</button> |
| 7 | 44 | DEF Institution | 31/12/2023 | Current | 1 | 2024-10-12 20:58:54 | 2024-10-12 20:59:18 | <button>View Details</button> |
| 3 | 5 | EFG Institution | 30/06/2023 | Current | 1 | 2024-10-08 18:12:35 | 2024-10-10 14:07:27 | <button>View Details</button> |
| 2 | 99 | STU Institution | 30/06/2024 | Current | 1 | 2024-10-08 12:01:12 | 2024-10-10 15:18:21 | <button>View Details</button> |
| 1 | 100 | TUV Institution | 30/06/2024 | Current | 1 | 2024-10-07 19:04:18 | 2024-10-10 15:21:33 | <button>View Details</button> |
| 4 | 97 | QRS Institution | 30/06/2024 | Current | 1 | 2024-10-09 10:12:01 | 2024-10-10 15:44:26 | <button>View Details</button> |
| 5 | 89 | IJK Institution | 30/06/2024 | Current | 1 | 2024-10-09 20:25:51 | 2024-10-13 15:35:18 | <button>View Details</button> |

Page Size: 100

1 to 7 of 7

Page 1 of 1

© 2024 Prudentia. All Rights Reserved.

| Review | Return | Firm | Reporting Date | State | Version | Created At | Updated At | Actions |
|--------|--------|-----------------|----------------|---------|---------|---------------------|---------------------|-------------------------------|
| | | | 13/10/2024 | | | | | |
| 6 | 64 | DEF Institution | 31/03/2024 | Current | 1 | 2024-10-11 12:25:59 | 2024-10-11 12:32:53 | <button>View Details</button> |
| 7 | 44 | DEF Institution | 31/12/2023 | Current | 1 | 2024-10-12 20:58:54 | 2024-10-12 20:59:18 | <button>View Details</button> |

PR Reviews Detail Page:

The PR reviews consist of 5 sections:

1. Capital Key Measures & Summary Comments
2. Liquidity Key Measures & Summary Comments
3. Investments Key Measures & Summary Comments
4. Credit Key Measures & Summary Comments
5. Final Summary Comments

The Key Measures are supplied via AG Grid tables. 5 quarters of data are provided along with comparative ratios for the previous quarter and previous year. The sectoral average of the key measure for the review period is also provided and an editable field that allows a comment on each individual key measure. Any data entered in the field is automatically saved.

The summary comments are provided via a textarea Input that also automatically saves data to the database on input.

If a logged in user is recorded as part of the Supervisor user group in the backend, they can “supervisor sign-off” the PR Review. Once this has been done, a logged in user recorded as part of the Senior_Supervisor user group in the backend can complete the review with “senior supervisor sign-off”. A supervisor must “supervisor sign-off” first before a senior supervisor can “senior supervisor sign-off” and complete the review.

PRUDENTIA

DEF Institution
2024-03-31

Liquidity Key Measures

| Key Measures | Current Qtr | Prior Qtr 1 | Prior Qtr 2 | Prior Qtr 3 | Prior Year | Δ% Prior Qtr 1 | Δ% Prior Year | Current Qtr Sector Average | Comments |
|----------------------------|-------------|-------------|-------------|-------------|------------|----------------|---------------|----------------------------|----------|
| Liquidity Ratio | 30.35 | 31.04 | 31.09 | 32.41 | | -2.22 | | 31.07 | |
| Short Term Liquidity Ratio | 18.98 | 19.45 | 19.61 | 20.98 | | -2.42 | | 19.66 | |

Liquidity Summary Comment:
testing

Note: Use the TestSupervisor and TestSeniorSupervisor user account to test out the 4-eyes sign-off. Usernames can be added to user groups in the Django Admin interface (logon as superuser to access).

Frontend Codebase Breakdown

The frontend end uses a Vite distribution of the React framework. Vite was chosen as it facilitates a working distribution of interacting packages that do not expose the codebase to vulnerabilities.

The code is written in React .jsx files primarily, with some .css and .html files.

The index.html is the page that renders the website which is fed by the main.jsx file which is in turn fed by the App.jsx file. The other jsx files are divided into folders based on their classification of being related to components, grids or pages.

Components:

Login.jsx, Logout.jsx, Register.jsx deal with user authentication with the backend.

FetchData.jsx covers fetch commands with the API.

Frontend security is covered by the authentication file, fetch file and the ProtectedRoute.jsx and NotFound.jsx files.

The ThemeContext.jsx and ThemeToggleIcon.jsx cover the management and activation of the light/dark theme throughout the website.

In website functionality such as PR Review creation is covered by CreateReview.jsx; addition of comments by PRCommentsSection.jsx and sign-off by PRReviewSignOffButtons.jsx.

TabStructure.jsx manages the switch between tabs on the Returns Details page. ReturnsFilter.jsx covers changing the input filters to select a different firm and reporting date.

FormatFunctions.jsx formats data in the grid tables.

PRReviewHeader.jsx fetches the firm name and reporting date for the PR Reviews Details page.

Pages & Grids:

The pages are fed to the App.jsx, the grids feed the pages.

- ContactPage.jsx is the contact page and has no grid feeders.
- InfoPage.jsx is the info page and has no grid feeders.

- The HomePage.jsx is the home dashboard pages and is fed by:
 - DashboardBottomLeftGrid.jsx – total loans chart
 - DashboardBottomRightGrid.jsx – total investments chart
 - DashboardTopLeftGrid.jsx – total assets chart
 - DashboardTopRightGrid.jsx – this is the return summary table
- The ReturnsListPage.jsx is the returns list page; it is fed by the ReturnsListGrid.jsx.
- The ReturnsDetailPage.jsx is the returns details page and is fed by:
 - IncomeExpenditureGrid.jsx
 - BalanceSheetGrid.jsx
 - InvestmentsDepositsGrid.jsx
 - CreditRiskGrid.jsx
 - LoanAnalysisGrid.jsx

These correspond to the 5 tabs that are managed by the TabStructure.jsx component.

- The PrReviewsPage.jsx is the PR reviews list page; it is fed by the PrReviewsGrid.jsx.
- The PrReviewsDetailsPage.jsx is the PR reviews details page and is fed by:
 - CapitalKeyMeasuresGrid.jsx
 - LiquidityKeyMeasuresGrid.jsx
 - InvestmentsKeyMeasuresGrid.jsx
 - CreditKeyMeasuresGrid.jsx

Note that the PRCommentsSection.jsx component handles the inclusion of the summary comment section for each key measure area.

Backend Features

The backend uses the Django framework and is built to function as a REST API. The frontend does not directly communicate with the database it does so via the API.

The backend handles authentication via JWT tokens. Password reset functionality is handled via the Django framework directly rather than the frontend. Links bring the user from the frontend to the backend for password reset functionality.

Swagger and Redocly was incorporated into the project to improve interaction and documentation of the API.

You must be logged into the backend as a superuser to interact directly with the API.

The link to the Prudentia backend's **Django Admin** section can be found here:

<https://prudentiaapi.onrender.com/admin/>

(see supplied password list file in UCDPA submission for a superuser username and password).

The link to the **Swagger** implementation for the API, accessible as a logged on superuser, is here: <https://prudentiaapi.onrender.com/swagger/>

The link to the **Redocly** automated API documentation for the project, accessible as a logged on superuser is here: <https://prudentiaapi.onrender.com/redoc/>

The Redoc API documentation can also be found here:

https://github.com/djonskanlyn/Prudentia/blob/main/prudentia_docs/prudentia_api_redoc.pdf

The REST Framework endpoints themselves can be accessed here:

- Data App: <https://prudentiaapi.onrender.com/api/data/>
- Key Measures App: <https://prudentiaapi.onrender.com/api/key-measures/>
- PR Reviews App: <https://prudentiaapi.onrender.com/api/pr-reviews/>

Automated Testing:

There are 12 automated tests facilitated by the tests.py file; these are divided among 5 classes:

1. Login test case (2 tests)
2. Register test case (3 tests)
3. Password reset test case (3 tests)
4. Swagger access test case (2 tests)
5. Redocly access test case (2 tests)

Backend Codebase Breakdown

The standard Django framework documentation can be found here:

<https://docs.djangoproject.com/en/5.1/>

The standard Django REST framework documentation can be found here:

<https://www.django-rest-framework.org>

The backend files can be found in the github here:

<https://github.com/djonskanlyn/Prudentia/tree/main/backend>

The required python libraries for the backend can be found in the requirements.txt file.

The backend comprises of 4 apps:

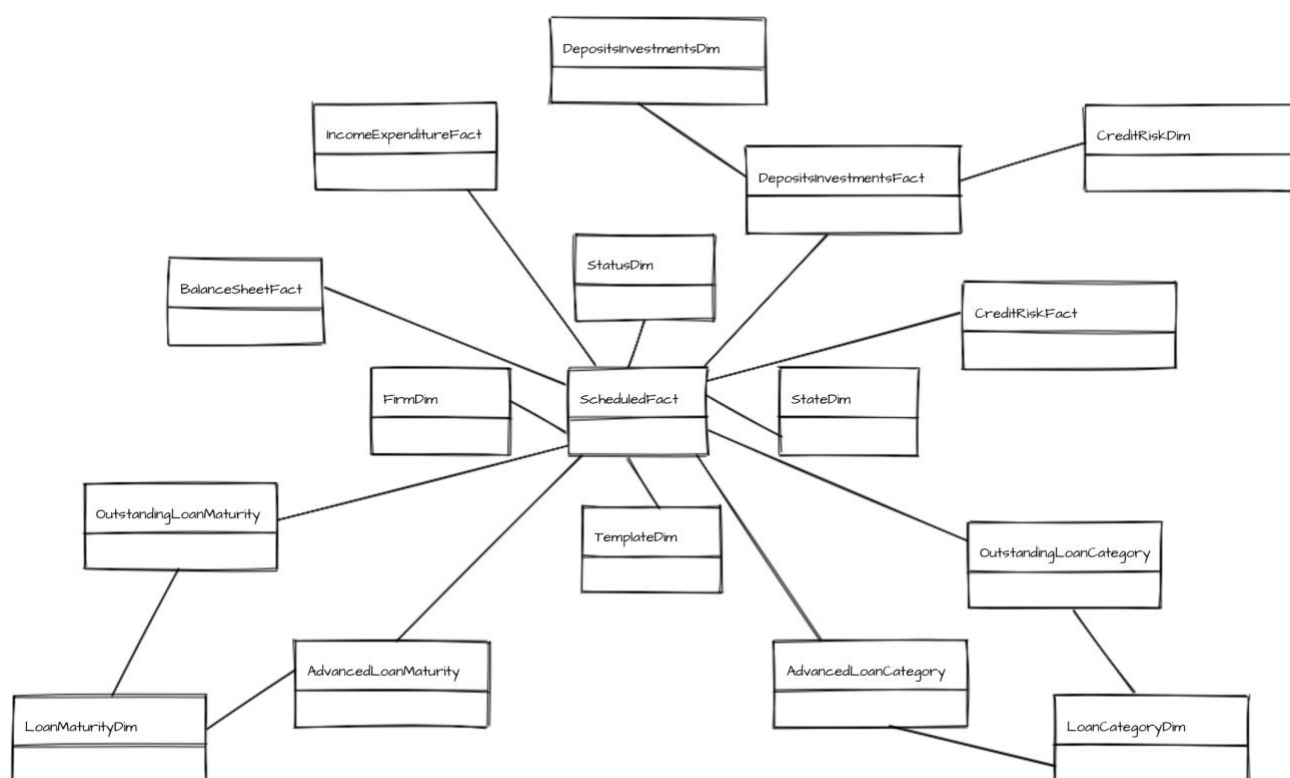
- **authapp:** handles authentication, login, logout, registration and related automated tests.
- **data:** this sets up the database tables that store the return data; these database tables are not written to; they supply the data to the frontend and the other apps.
- **key-measures:** this handles the calculation of key measures from data supplied in the database. There is a signals.py set-up to calculate new key measures when data is added to the data database tables. However, for the purposes of this project a python command file calculated both the key measures and sectoral averages. This can be within the commands folder within the management folder of the key-measures app.
- **pr-reviews:** this app has some tables that are written too on the creation of pr-reviews and the addition of comments to the PR Review.

Database

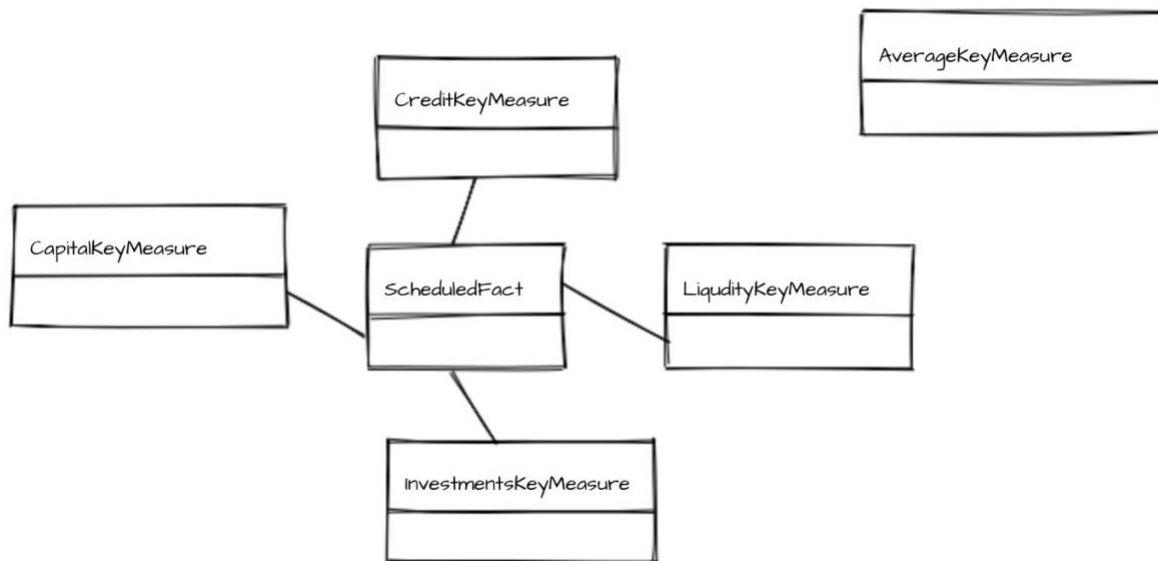
The data in the database and the structure of the tables were designed based on public returns published by a particular financial institution. Other public sources were used to refine dimensions. The proportions for this financial institution were used with Chatgpt to generate variant data for 20 made-up firms with 5 quarters worth of data each. The data for a return is entirely made-up. Using Chatgpt was not entirely successful, however it was an excellent starting point for which the work was refined in Excel (balancing the data points, dealing with rounding issues etc.). I have included the files and the Chatgpt conversation in the project submission (not github). Key measures were logical extractions from the return data, additionally not all key measures are entirely accurate (eg. liquidity) but suffice for the purposes of the project. Non-serious comments were added, this was deliberate, the purpose was to show the facility to comment, not what would be commented.

The database can be recreated by migrating the models and using the datadump.json file to reload the data. Originally the tables were populated by uploading csv files to PostgreSQL database directly. Key measures and sector averages were calculated by management commands in the key measures app.

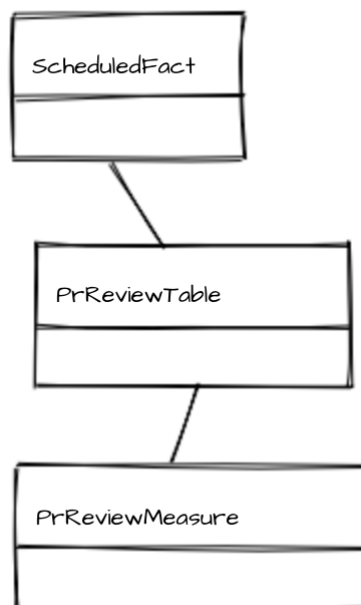
A map of the database structure for the data app is as follows:



A map of the database structure for the key measures app is as follows:



A map of the database structure for the pr reviews app is as follows:



The standard auth tables generated by Django are not described here, see the standard Django documentation.

Development Diary

Coding commenced on 8th September 2024. Prior to this the concept had been developed, and the aesthetics of the webpage prepared in HTML and CSS. See presentation.

- 3rd September 2024: Initial commit of project to github.
- 8th September 2024: Django backend, REST framework, React frontend was set-up; and a working connection from frontend to backend using a todo list tutorial
- 10th September 2024: Got login and register functional from frontend to backend working.
- 11th September 2024: Got Swagger and Redoc set up in the project.
- 12th September 2024: Styled frontend as designed, added login protections to pages.
- 13th September 2024: Refined frontend aesthetics, further worked on theme switch light to dark; started styling the AG Grids for the Return Lists.
- 15th September 2024: Moved off Sqlite3 having set up PostgreSQL database. Defined data apps models.py tables.
- 16th September 2024: Ran database migrations for data app; uploaded data to database using csvs, created datadump json to redeploy database if required. Worked on the serializers, views and urls python files for data app.
- 18th September 2024: Returns List grid connected backend api.
- 19th September 2024: Started on the Returns Details section. Income and Expenditure and Balance Sheet grids styled and connected.
- 20th September 2024: Deposits and Investments tab styled, credit risk disclosures styled, loan analysis styled and all connected.
- 21st September 2024: Implementation of dynamic return id to allow inputs to switch firm and reporting date for returns details. More styling, tab switching and local storage implementation of inputs.
- 27th September 2024: set up of investment, capital and liquidity measures in key-measures app.

- 28th September 2024: set up credit key measures, worked on the serializer, views and urls files for the key-measures app. Created some views as endpoints.
- 29th September 2024: more development on endpoints, added calculate averages functionality, created the working tables for the pr-review app.
- 30th September 2024: setup so pr review can be created from frontend.
- 1st October 2024: Got the capital key measures grid implemented.
- 5th October 2024: Worked on testing the website in production, got webservice and static site working. Testing from now on will involve testing on local, then testing on production.
- 7th October 2024: protected backend with authentication requirements.
- 8th October 2024: got password reset functioning.
- 9th October 2024: commenting functionality on key measures
- 10th October 2024: sign-off functionality implemented
- 11th October 2024: finish dashboard and implement automated testing with tests.py
- 12th October 2024: info page updated, documentation
- 13th October 2024: further documentation
- 14th October 2024: project finalisation.

Reset-up Instructions

Run in development on local:

Assumptions:

- PostgreSQL database still running on Render
- Frontend to run at localhost:3000
- Backend to run at localhost:8000

4 files need to be adjusted to run in development in local:

1. Backend -> Backend -> settings.py
 - Comment in line 29: `DEBUG = True`
 - Comment out line 30: `DEBUG = False`
 - Comment in line 32: `FRONTEND_URL = 'http://localhost:3000/'`
 - Comment in line 33: `FRONTEND_URL = 'https://prudentia.onrender.com/'`
2. Frontend -> src -> components -> Login.jsx
 - Comment in line 24: `const response = await axios.post('http://localhost:8000/api/token/', formData);`
 - Comment out line 26: `const response = await axios.post(`${import.meta.env.VITE_PRUDENTIA_API_BASE_URL}token/`, formData);`
 - Comment in line 73: `<p>Forgot password? Reset Password</p>`
 - Comment out line 74: `<p>Forgot password? Reset Password</p>`
3. Frontend -> src -> components -> Register.jsx
 - Comment in line 24: `const response = await axios.post('http://localhost:8000/api/register/', formData);`
 - Comment out line 26: `const response = await axios.post(`${import.meta.env.VITE_PRUDENTIA_API_BASE_URL}register/`, formData);`
4. Frontend -> src -> components -> FetchData.jsx
 - Comment out line 5: `baseUrl:`
`import.meta.env.VITE_PRUDENTIA_API_BASE_URL,`
 - Comment in line 7: `baseUrl: 'http://localhost:8000/api/'`

Environment Variables: (Details in project submission not on Github)

The following environment variables need to be set up to run on local:

1. FRAMEWORKS_PROJECT_EMAIL_PASS
2. FRAMEWORKS_PROJECT_EMAIL_USER
3. PRUDENTIA_DATABASE_URL
4. PRUDENTIA_SECRET_KEY

PostgreSQL Database Set-up:

Info

General

Name
A unique name for your database.

prudentia-postgresql

Edit

Created
a month ago

Status
✓ Available

PostgreSQL Version
16

Region
Frankfurt (EU Central)

Read Replica
+ Add Read Replica

Storage
7% used out of 1 GB

Datadog API Key
+ Add Datadog API Key

Connections

Hostname
An internal hostname used by your Render services.

dpg-crf9cadds78s73cj7h50-a

Port

5432

Database

prudentia_postgresql

Username

prudentia_postgresql_user

Password

Internal Database URL

External Database URL

PSQL Command

Note: The backend folder structure has a folder named 'dumpdata' that contains the initial returns data to populate the database. The key measures and sector averages will need to be calculated by running the command statements in the key-measures app under the management folder.

Web Service Set-up of Backend:

The set-up instructions for the static site on Render are as follows:

Settings

General

Name
A unique name for your Web Service.

PrudentiaAPI

Edit

Region
Your services in the same [region](#) can communicate over a [private network](#).

Frankfurt (EU Central)

Instance Type

Starter

0.5 CPU

512 MB

Update

Build & Deploy

Repository
The repository used for your Web Service.

https://github.com/djonskanlyn/Prudentia

Edit

Branch
The Git branch to build and deploy.

main

Edit

Root Directory Optional
If set, Render runs commands from this directory instead of the repository root. Additionally, code changes outside of this directory do not trigger an auto-deploy. Most commonly used with a [monorepo](#).

backend

Edit

Build Filters
Include or ignore specific paths in your repo when determining whether to trigger an auto-deploy. [Learn more](#).
Paths are relative to your repo's root directory.

Included Paths
Changes that match these paths will trigger a new build.

+ Add Included Path

Ignored Paths
Changes that match these paths will not trigger a new build.

+ Add Ignored Path

Edit

Build Command
 Render runs this command to build your app before each deploy.

backend/ \$ pip install -r requirements.txt
 [Edit](#)

Pre-Deploy Command Optional
 Render runs this command before the start command. Useful for database migrations and static asset uploads.

backend/ \$
 [Edit](#)

Start Command
 Render runs this command to start your app with each deploy.

backend/ \$ gunicorn backend.wsgi
 [Edit](#)

Auto-Deploy
 By default, Render automatically deploys your service whenever you update its code or configuration. Disable to handle deploys manually. [Learn more](#).

Yes
 [Edit](#)

Deploy hook
 Your private URL to trigger a deploy for this server. Remember to keep this a secret.

.....
 [Regenerate hook](#)

Environment

Environment Variables
 Set environment-specific config and secrets (such as API keys), then read those values from your code. [Learn more](#).
 [Create environment group](#)

| Key | Value |
|-------------------------------|----------------------------|
| FRAMEWORKS_PROJECT_EMAIL_PASS | Edit |
| FRAMEWORKS_PROJECT_EMAIL_USER | Edit |
| PRUDENTIA_DATABASE_URL | Edit |
| PRUDENTIA_SECRET_KEY | Edit |

[Edit](#)

Static Site Set-up of Frontend:

Settings

General

Name
 A unique name for your Static Site.

Prudentia

[Edit](#)

Build & Deploy

Repository

The repository used for your Static Site.

https://github.com/djonskanlyn/Prudentia

Edit

Branch

The Git branch to build and deploy.

main

Edit

Root Directory Optional

If set, Render runs commands from this directory instead of the repository root. Additionally, code changes outside of this directory do not trigger an auto-deploy. Most commonly used with a [monorepo](#).

frontend

Edit

Build Filters

Include or ignore specific paths in your repo when determining whether to trigger an auto-deploy. [Learn more](#).

Paths are relative to your repo's root directory.

Included Paths

Changes that match these paths will trigger a new build.

+ Add Included Path

Ignored Paths

Changes that match these paths will not trigger a new build.

+ Add Ignored Path

Edit

Build Command

Render runs this command to build your app before each deploy.

frontend/ \$ npm install; npm run build

Edit

Publish directory

The relative path of the directory containing built assets to publish. Examples: `./build`, `dist` and `frontend/build`.

frontend/ dist

Edit

Auto-Deploy

By default, Render automatically deploys your service whenever you update its code or configuration. Disable to handle deploys manually. [Learn more](#).

Yes

Edit

Deploy hook

Your private URL to trigger a deploy for this server. Remember to keep this a secret.

.....

Regenerate hook

Environment

Environment Variables

Set environment-specific config and secrets (such as API keys), then read those values from your code. [Learn more](#).

Create environment group

| Key | Value |
|-----------------------------|-------|
| VITE_PRUDENTIA_API_BASE_URL | |

Edit