

Project Documentation

EXPENSIO

(Web project using Django & HTML/CSS)

TEAM MEMBER	E-MAIL ADDRESS	MATRIKEL Nr.
Alexandra Raluca Jelea	alexandra.jelea@uni.li	FS190340
Kaoru Steinböck	kaoru.steinboeck@uni.li	
Gerhard Nägele	gerhard.naegele@uni.li	FS200004

Submission Date: 04.06.2020

Content

Project	3
Ideas Description	3
Mockup / Paper Prototypes	3
System Description.....	4
External Packages/Libraries	4
Design Decisions.....	5
Database.....	6
Data Model in Django.....	7
Code.....	9
Structure.....	9
URL's	10
Views	11
Version Control System	13
Contributions.....	14
Additional Information	14
Challenges and Findings	17
Start/Run Project.....	18
User Manual	19
Main Page.....	19
Register User	20
Login	20
Dashboard	22
Transactions	23
Budget	25
Category	27
Payment	28
User Profile.....	30
Upload CSV file into EXPENSIO.....	31

Project

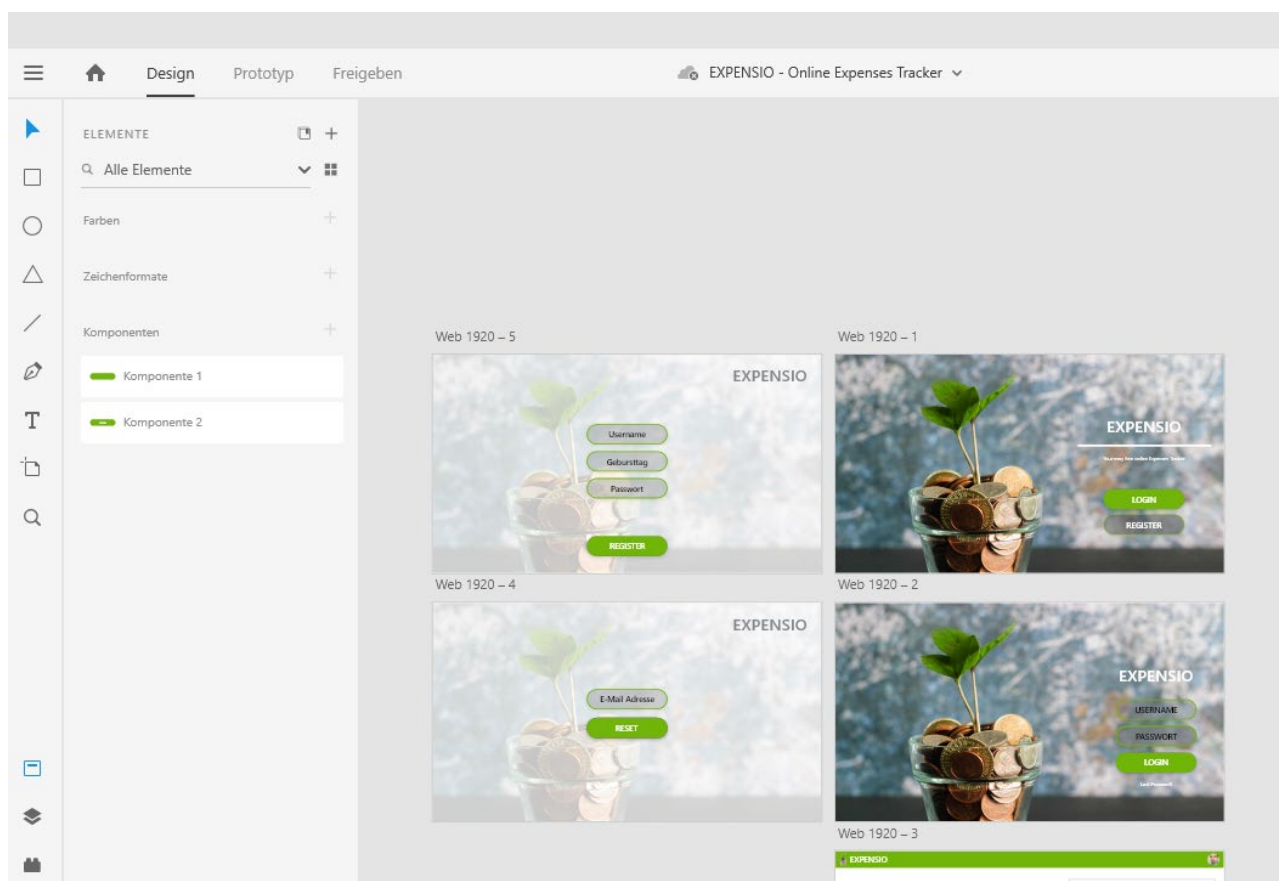
Ideas Description

The basic idea is to program a simple expense tracker using Django, with which you can record expenses, assign categories, create a budget for different expenses and a simple csv export and import function as well as overviews via charts. The App should be responsive and follow an intuitive usage concept.

- Tracking Expenses (CRUD)
- Import/Export Function
- Create Budget (CRUD)
- Tags or Categories (CRUD)
- User Management (Multiuser compatible)

Mockup / Paper Prototypes

We used AdobeXD and Paper Prototypes for our Mockups to design the basic functionalities.



System Description

Used Technologies:

Programming Language	Python, HTML, JavaScript
Webframework	Django V. 3.0.6
Mockup Tool	AdobeXD
IDE / Editor	PyCharm / Emacs
Database	SQLite V. 3
OR-Mapper	Django-ORM
Version Control System	Github, Git Version 2.26.0
Documentation	Docstrings Sphinx's
Project Management	Github Kanban Board

External Packages/Libraries

django-crispy-forms V. 1.9.1: <https://github.com/django-crispy-forms/django-crispy-forms>

Bootstrap V. 4.4.1: <https://getbootstrap.com/docs/4.4/getting-started/introduction/>

Chart.js V. 2.9.3: <https://www.chartjs.org/docs/latest/getting-started/installation.html>

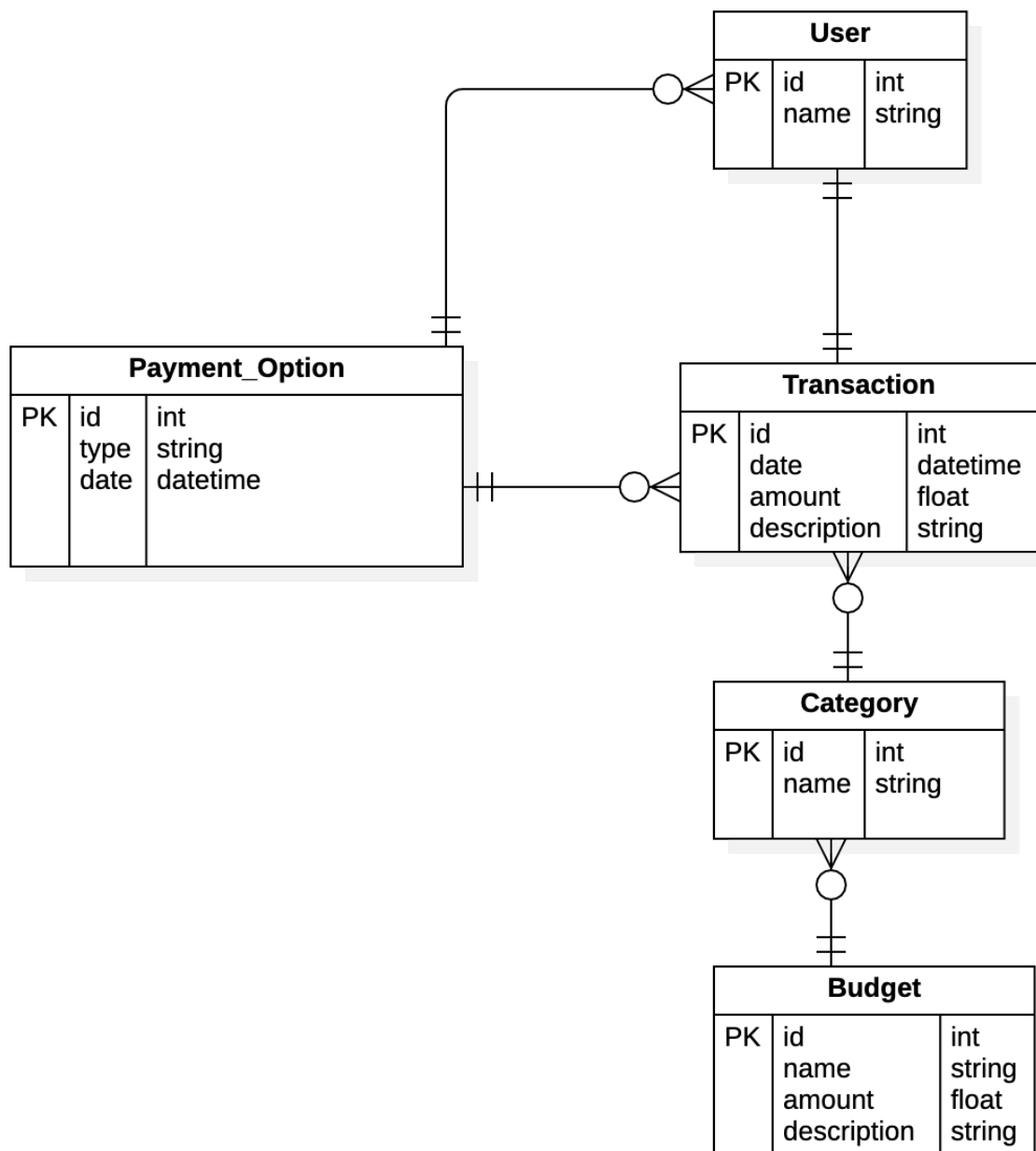
Font Awesome: <https://fontawesome.com/>

Design Decisions

1. Every user can only see his own data. That includes all budgets, transactions, and categories.
2. To see the functions, a user has to be logged in. If he wants to open a page with a required login status, the user is redirected to the login page.
3. For this project, the company data, contact person for data protection, imprint etc. are not mentioned.
4. The app uses only CHF as Currency.
5. CSV Upload is only possible in a predefined format. This format gets explained on the upload Page.
6. Payment types are given.
- 7.

Database

ER Diagram – without Foreign Keys. FK are visible in the Data Model Section of this Documentation.



The ER Diagramm.mdj File and a png of the ER Diagram are available on Github:

SS-20-Information-Systems-Development---Expense-Tracker/doc/**Diagrams/**

Data Model in Django

Transaction Data Model:

```
class Transaction_Expense(models.Model):
    date = models.DateTimeField()
    amount = models.DecimalField(max_digits=19, decimal_places=2)
    description = models.CharField(max_length=200, default="")
    user = models.ForeignKey("auth.User", on_delete=models.CASCADE)
    category = models.ForeignKey(Category, on_delete=models.CASCADE, null=True)
    payment = models.ForeignKey(Payment, on_delete=models.CASCADE, null=True)
```

Payment_Option Data Model:

```
class Payment(models.Model):
    VISA = "VISA"
    MASTERCARD = "MASTERCARD"
    BANK = "BANK"
    CRYPTO = "CRYPTO"
    CASH = "CASH"
    PAYMENT_TYPE = [
        (CASH, "Cash"),
        (VISA, "Visa"),
        (MASTERCARD, "Mastercard"),
        (BANK, "Bank"),
        (CRYPTO, "Crypto"),
    ]
    type = models.CharField(max_length=50, choices=PAYMENT_TYPE, default=CASH, )
    date = models.DateTimeField()
    description = models.CharField(max_length=100, default="")
    user = models.ForeignKey("auth.User", on_delete=models.CASCADE)

    def __str__(self):
        return f"Type: {self.type}, Description: {self.description}"
```

User Data Model:

```
class UserProfileSettingConfig(models.Model):
    user = models.OneToOneField(User, on_delete=models.CASCADE)
    description = models.CharField(max_length=100, default='')
    city = models.CharField(max_length=200, default='')
    phone = models.IntegerField(default=0)
```

```
def create_user_profile(sender, instance, created, **kwargs):
    if created:
        UserProfileSettingConfig.objects.create(user=instance)
```

```
post_save.connect(create_user_profile, sender=User)
```

Category Data Model:

```
class Category(models.Model):
    name = models.CharField(max_length=100)
    created_date = models.DateTimeField(auto_now_add=True)
    user = models.ForeignKey("auth.User", on_delete=models.CASCADE)

    """
    Returns self.name instead of object.object
    """

    def __str__(self):
        return str(self.name)

    def get_absolute_url(self):
        return reverse("category-detail", kwargs={"pk": self.pk})
```

Budget Data Model:

```
class Budget(models.Model):
    name = models.CharField(max_length=50)
    amount = models.FloatField(null=True)
    description = models.CharField(max_length=200)

    created_date = models.DateTimeField(auto_now_add=True)
    user = models.ForeignKey("auth.User", on_delete=models.CASCADE)
    category = models.ForeignKey(Category, on_delete=models.CASCADE, null=True)

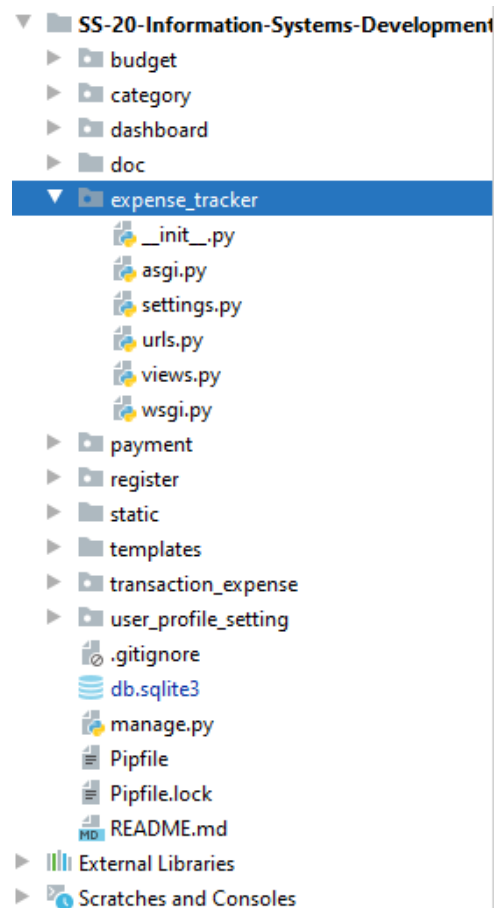
    def __str__(self):
        return str(self.name)
```

Code

This project should be following the pep8 formatting standard, which should be carried out by the used editors PEP 8.

EXPENSIO uses several APP's to provide the different functionalities.

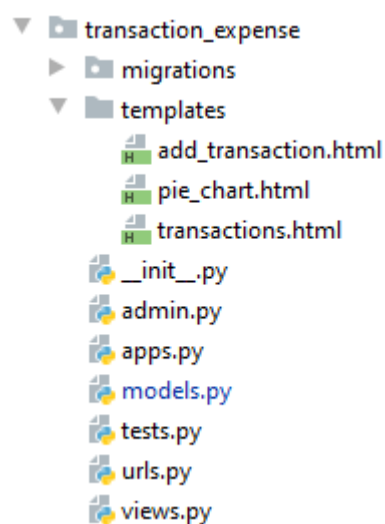
Structure



The folder expense_tracker holds the settings, urls and views python files.

EXPENSIO App's are:

- budget
- category
- dashboard
- payment
- register
- transaction_expense
- user_profile_setting



APP's Structure

Every APP uses their own templates folder to store all templates, has a urls.py file with all the connections between the views with templates and the URL path.

URL's

In the main urls.py file in the folder expense_tracker is the include statement for the urls.py files in the different APP's.

Urls.py of expense_tracker

```
urlpatterns = [
    path("", views.welcome, name="welcome"),
    path("dashboard/", include("dashboard.urls")),
    path("category/", include("category.urls")),
    path("budget/", include("budget.urls")),
    path("admin/", admin.site.urls),
    path("register/", register_view.register, name="register"),
    path(
        "login/",
        auth_view.LoginView.as_view(template_name="register/login.html"),
        name="login",
    ),
    path(
        "logout/",
        auth_view.LogoutView.as_view(template_name="register/logout.html"),
        name="logout",
    ),
    path("", include("django.contrib.auth.urls")),
    path("transaction/", include("transaction_expense.urls")),
    path("payment/", include("payment.urls")),
    path("profile/", include("user_profile_setting.urls")),
    path("profile/", include("user_profile_setting.urls")),
]
```

Example of urls.py file in an APP - expense_tracker

```
urlpatterns = [
    path("", login_required(TransactionExpenseListView.as_view()), name="transactions"),
    path(
        "add/",
        login_required(TransactionExpenseCreateView.as_view()),
        name="add_transaction",
    ),
    path(
        "<int:pk>/edit",
        login_required(TransactionExpenseEditView.as_view()),
        name="edit_transaction",
    ),
    path("<int:pk>/delete", login_required(TransactionExpenseDeleteView.as_view())),
    path(
        "pie_chart/",
        login_required(views.transaction_expense_pie_chart),
        name="pie-chart",
    ),
]
```

Views

The views.py file hold the Code for the view generation and Code to provide the functions for the view generation or provide the needed data in the return.

Example of a view Class generation

```
class UserProfileSettingConfigView(TemplateView):
    template_name = "user_profile_detail.html"

    def form_valid(self, form):
        form_to_save = form.save(commit=False)
        form_to_save.user = self.request.user
        return super(UserProfileSettingConfigView, self).form_valid(form)
```

Example of csv import function

```
# https://docs.djangoproject.com/en/3.0/topics/db/queries/#creating-objects
# https://stackoverflow.com/questions/8636760/parsing-a-datetime-string-into-a-django-datetimefield
# https://www.foxinfo.tech.in/2018/09/python-how-to-skip-first-line-in-csv.html
def transaction_upload(request):
    template = "user_profile_detail.html"
    prompt = {"order": "Date, Amount, User, Payment, Description, Category"}

    # h
    # h
    if request.method == "GET":
        return render(request, template, prompt)

    csv_file = request.FILES["file"]

    data_set = csv_file.read().decode("UTF-8")
    user = request.user

    csv_data = csv.reader(data_set.split("\n"), delimiter=",", quotechar='"')
    next(csv_data)
    for column in csv_data:
        if len(column) is not 0:
            # print('column: ', column)
            csv_date = parse_datetime(column[0])
            csv_amount = column[1]
            csv_payment_data = column[3]
            csv_description = column[4]
            csv_category = column[5]
            user = User.objects.get(username=user)
            category = Category.objects.get(name=csv_category, user=user)
            csv_payment_type, csv_payment_description = get_payment(csv_payment_data)
            payment = Payment.objects.get(type=csv_payment_type, user=user)
            transaction = Transaction_Expense(
                date=csv_date,
                amount=csv_amount,
                user=user,
                category=category,
                payment=payment,
                description=csv_description,
            )
            transaction.save()
        context = {}
    return render(request, template, context)
```

Example of login function

```
# https://www.youtube.com/watch?v=z4LfVsb\_7MA - for creating the login
# https://www.youtube.com/watch?v=3aVqWaljqS4
@login_required
def profile(request, *args, **kwargs):
    return render(request, "profile.html")
```

Version Control System

The source code and version control was realized with Github.

The repository was created by Kaoru Steinböck. Github was used by all team members to push their own code or get the work of others. Only the Master Branch was used.

<https://github.com/kaoru-uni/SS-20-Information-Systems-Development---Expense-Tracker>

kaoru-uni / SS-20-Information-Systems-Development---Expense-Tracker
Unwatch 3
Star 1
Fork 0

Code
Issues 0
Pull requests 0
Actions
Projects 1
Wiki
Security 0
Insights

No description, website, or topics provided.

117 commits
1 branch
0 packages
0 releases
3 contributors

Branch: master
New pull request
Create new file
Upload files
Find file
Clone or download

kaoru-uni updated: ER diagram to display our changes.		Latest commit 17dbc4c 12 minutes ago
budget	Import csv file	4 hours ago
category	Import csv file	4 hours ago
dashboard	Import csv file	4 hours ago
doc	updated: ER diagram to display our changes.	12 minutes ago
expense_tracker	changed: changed dashboard url to root /.	6 hours ago
payment	Import csv file	4 hours ago
register	Formated float number Amount to 2 decimal numbers.	5 hours ago
static/expense_tracker	Added Welcome Page with Background Image.	6 hours ago
templates	changed: transaction order by date. only show menu if logged in.	2 hours ago
transaction_expense	changed: transaction order by date. only show menu if logged in.	2 hours ago
user_profile_setting	Import csv file	4 hours ago
.gitignore	Payment	2 months ago
Pipfile	Deleted files	last month
Pipfile.lock	added: sphinx documentation	4 days ago
README.md	Update README.md	yesterday
db.sqlite3	Changed Awesomefond icon color to the new color scheme color:#03ab30	5 hours ago
manage.py	added: django project setup. project setup readme.	2 months ago

Contributions

Apr 5, 2020 – Jun 3, 2020

Contributions: Commits ▾

Contributions to master, excluding merge commits



Additional Information

We will push our SQLite database into the git repository for easier debugging between the developers. If something goes wrong we can still go back to older committed databases. If we experience any problems we will remove the database from the repository.

We are aware that a binary file or database should not be committed. We are making an exception in this case.

README

github.com/kaoru-uni/SS-20-Information-Systems-Development---Expense-Tracker 🔍 ☆ |

manage.py added: django project setup: project setup readme. 2 months ago

README.md

Expense Tracker

this django project will be an 'Expense Tracker'

Project Team Member:

- Alexandra Jelea
- Kaoru Steinböck
- Gerhard Nägele

General information

this project should be following the pep8 formatting standard, which should be carried out by the used editors [PEP 8](#)

Installation

the following commands were run to install this project:

Setup the dev environment

1. to install or activate the virtualenv we are using [Pipenv](#)

```
pipenv shell
```

2. install django. In this case the latest version will be installed. `pipenv install django`
3. create the project. `django-admin startproject 'expense_tracker' .`

Start the project

1. to run or start the project `python manage.py runserver`
2. open the following website `http://localhost:8000/`

to run the project

please run the following commands to start the project

1. to run the virtualenv `pipenv shell`
2. to install all dependencies `pipenv install`
3. to run or start the project `python manage.py runserver`
4. open the following website `http://localhost:8000/`

Additional Information

We will push our SQLite database into the git repository for easier debugging between the developers. If something goes wrong we can still go back to older committed databases. If we experience any problems we will remove the database from the repository.

We are aware that a binary file or database should not be committed. We are making an exception in this case.

External Packages

- [django-crispy-forms](#)
- [Bootstrap](#)
- [Chart.js](#)
- [Font Awesome](#)

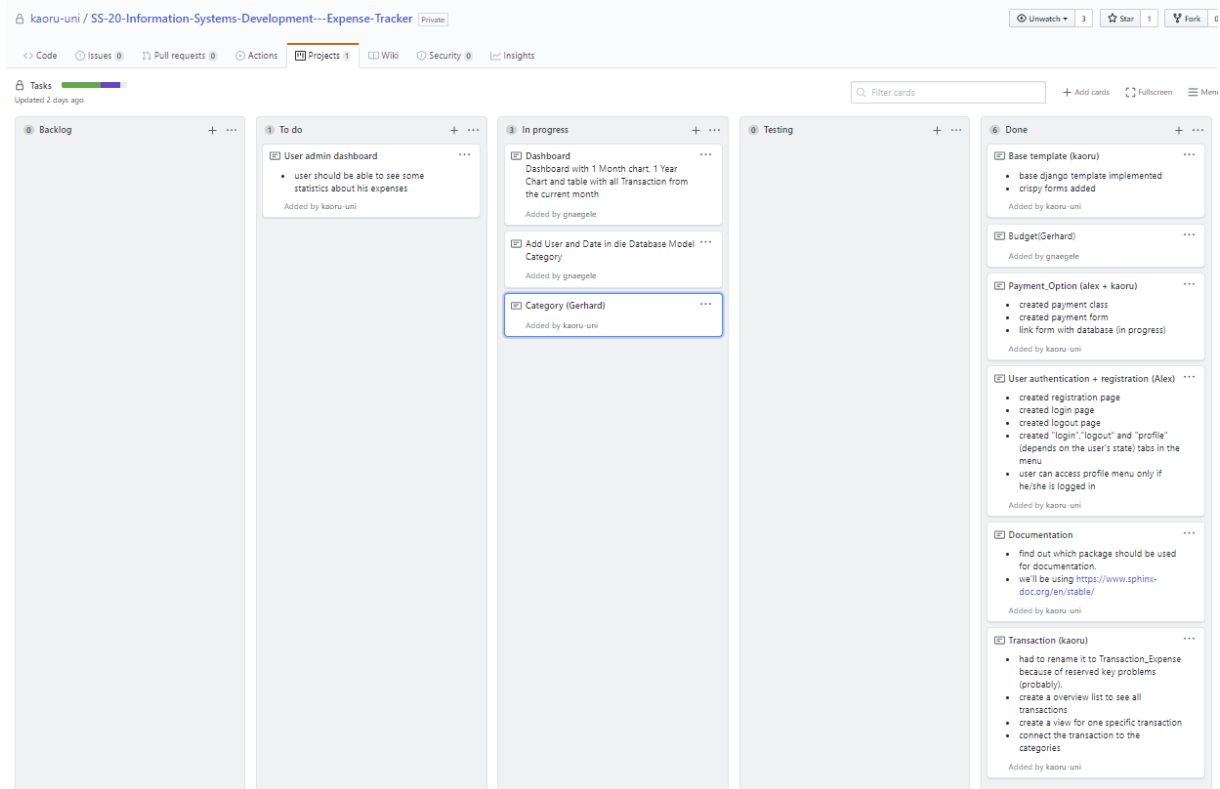
Sources

the following sources have been used in general to create this project

- [Django documentation](#)
- [Django for Beginners: Build websites with Python and Django \(English Edition\)](#)

Project-Management Function of Github

To coordinate the work between the Group members and keep the actual Project status available to all team members, we used the Github Function Project with a Kanban Board like functionality.



Challenges and Findings

CHALLENGES:

- Python was a new language for some of us
- Understand the View Concept
- VCS – we had some issues with merge conflicts. (These do not appear with git but with the PyCharm build in VCS Module)

FINDINGS:

- Django has a big Community. Solution for every problem we encounter.
- In Contrast to other Programming Languages Django Documentation is easier to read/understand.
- Even older Solutions from Stackoverflow worked
- Django does most of the Job (ORM, Return Values, Forms,...)

Start/Run Project

Prerequisites

The computer must have Python version 3.8 and Pipenv installed.

Pipenv: <https://pipenv.pypa.io/en/latest/>

Python Version 3.8 : <https://www.python.org/downloads/>

Tested Environments

Windows 10 Version 1903/1909 and macOS Catalina 10.15,

Setup the dev environment

1. to install or activate the virtualenv we are using Pipenv
2. type
cmd: ***pipenv shell***
in command line (in the project directory)
3. install django. In this case the latest version will be installed.
cmd: ***pipenv install django***
4. create the project.
cmd: ***django-admin startproject 'expense_tracker'***

Start the project

1. To run or start the
cmd: ***project python manage.py runserver***
2. Open the following website
Browser: ***<http://localhost:8000/>***

Run the Project

Please run the following commands to start the project

1. to run the virtualenv
cmd: ***pipenv shell***
2. to install all dependencies
cmd: ***pipenv install***
3. To run or start the
cmd: ***project python manage.py runserver***
4. Open the following website
Browser: ***<http://localhost:8000/>***

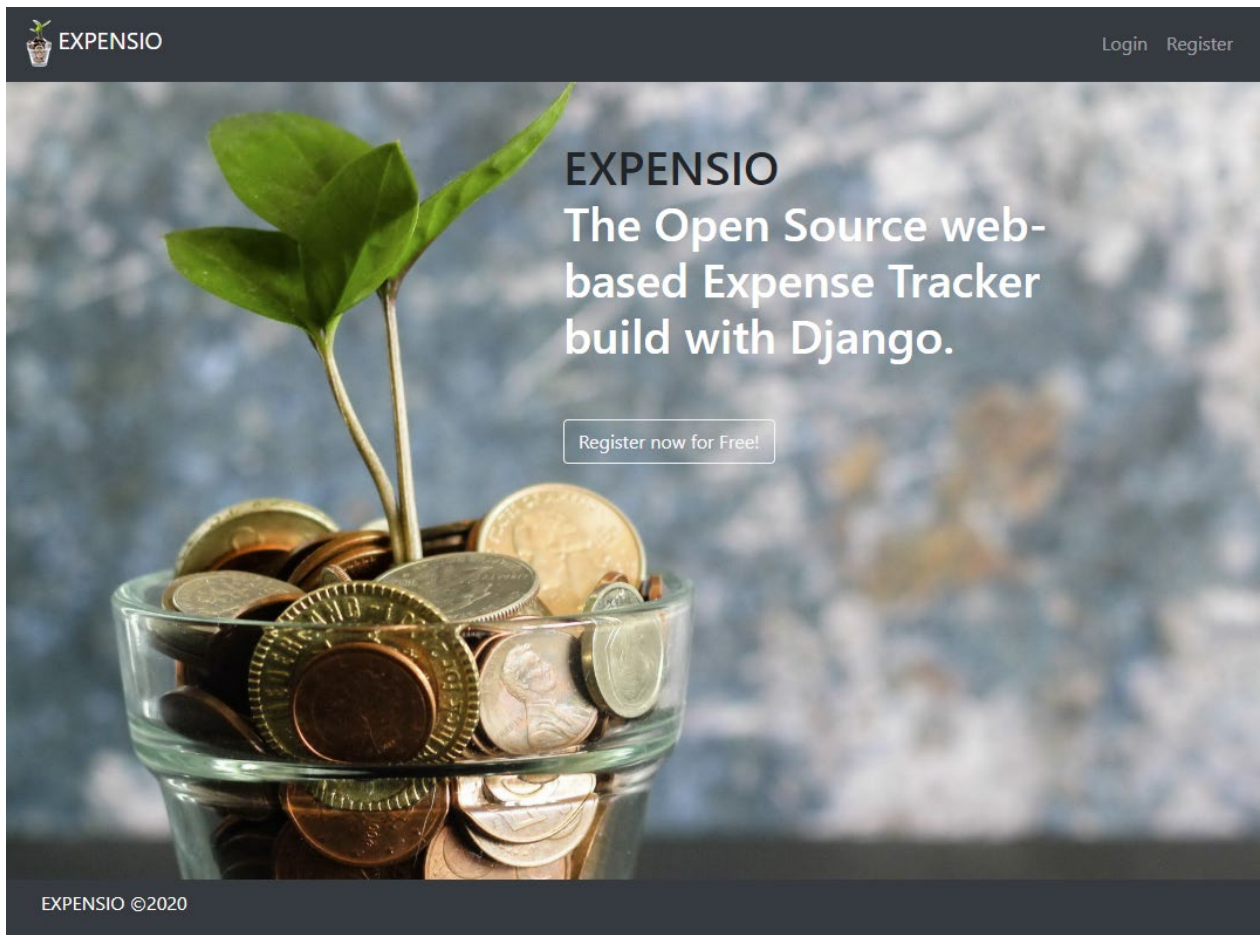
```
Gerhard@DESKTOP-GD50IQ7 MINGW64 ~/PycharmProjects/SS-20-Information-Systems-Development---Expense-Tracker (master)
$ python manage.py runserver
Watching for file changes with StatReloader
[03/Jun/2020 20:16:24] "GET / HTTP/1.1" 200 4321
[03/Jun/2020 20:16:24] "GET /static/expense_tracker/logo_small.png HTTP/1.1" 304 0
[03/Jun/2020 20:16:24] "GET /static/expense_tracker/Background_Small.jpg HTTP/1.1" 304 0
```

User Manual

Welcome to EXPENSIO the open source expense tracker. To use EXPENSIO please create a user. This can be done under Register.


To use the functionality of EXPENSIO the user has to be logged in. All data entered by a user is only visible to the user.

Main Page



Register User

To register you have to enter a Username, E-Mail Address and password. Then click on the **green** register Button.

 EXPENSIO [Dashboard](#) [Transactions](#) [Budget](#) [Category](#) [Payment](#) [Login](#) [Register](#)

Username*

Required. 150 characters or fewer. Letters, digits and @/./+/-/_ only.

Email*

Password*

- Your password can't be too similar to your other personal information.
- Your password must contain at least 8 characters.
- Your password can't be a commonly used password.
- Your password can't be entirely numeric.


Password confirmation*

Enter the same password as before, for verification.

Register

Login

To login you have to enter a Username and password. Then click on the **green** Login Button.

 EXPENSIO [Dashboard](#) [Transactions](#) [Budget](#) [Category](#) [Payment](#) [Login](#) [Register](#)

Username*

Password*

Don't have an account? Create one [here](#)

Login

Logout

When you logout the only available actions are to Register or Login.

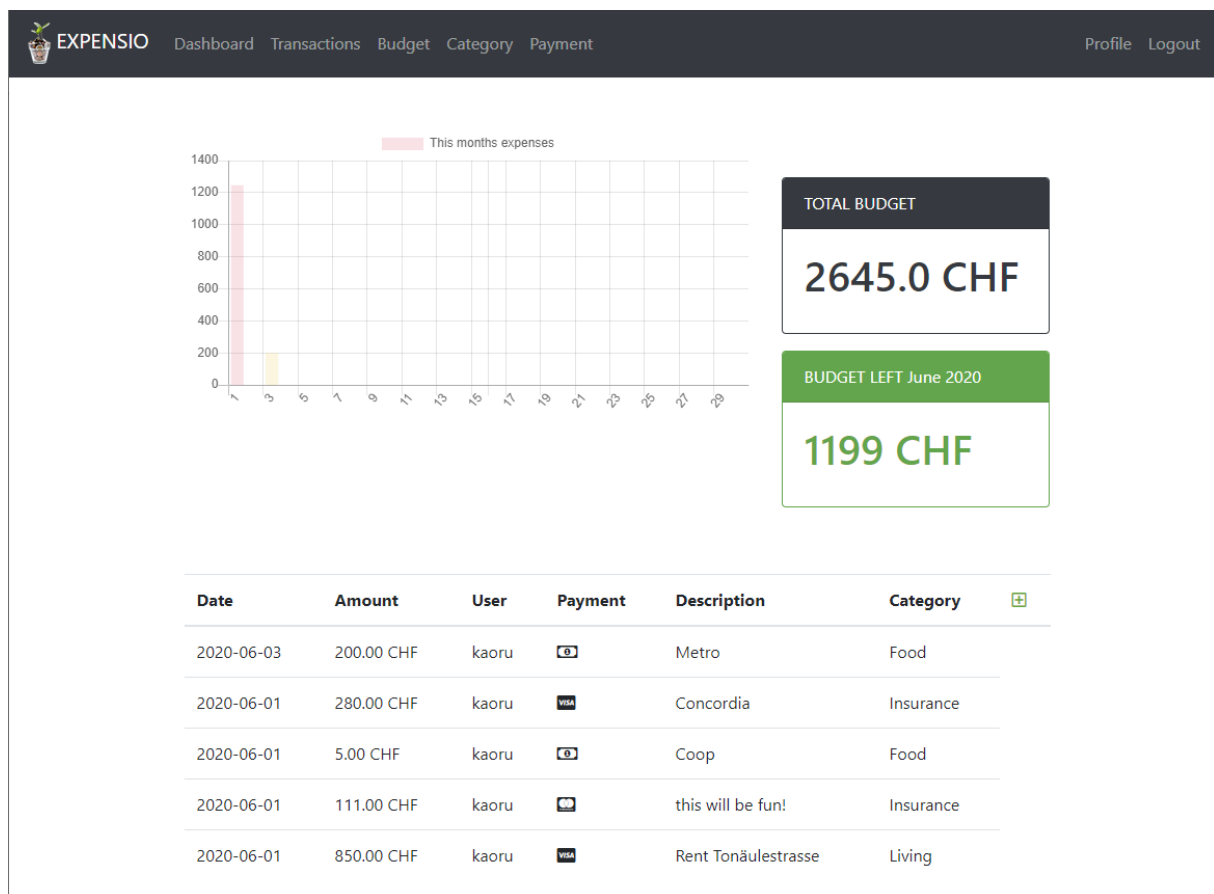


You have successfully log out.



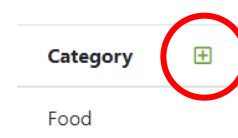
Dashboard

The Dashboard is the main information page. Here all expenses of the current month can be seen.




On the right side is the total budget for this month visible and the budget which is left for this month. The green color signals the user, that there is still budget left for expenses. Should the budget be exhausted, the color changes to red, this indicates that the expenditure should be limited.

To **add a new Expense** over the plus function on the dashboard. Click on the green Plus on the right corner on the Dashboard Table.
























Transactions


On the main page of the transaction App all transactions are shown.


 EXPENSIO [Dashboard](#) [Transactions](#) [Budget](#) [Category](#) [Payment](#) [Profile](#) [Logout](#)

[Add Expense](#) [Show Barchart](#)

Date	Amount	User	Payment	Description	Category	Edit	Delete
2020-06-01	280.00 CHF	kaoru		Concordia	Insurance		
2020-06-01	5.00 CHF	kaoru		Coop	Food		
2020-06-01	111.00 CHF	kaoru		this will be fun!	Insurance		
2020-06-01	850.00 CHF	kaoru		Rent Tonäulestrasse	Living		
2020-05-25	120.00 CHF	kaoru		Zurich Insurance	Insurance		
2020-06-03	200.00 CHF	kaoru		Metro	Food		
2020-06-03	2000.00 CHF	kaoru		Dryer	Living		

Edit transaction

To edit a transaction, click on the  edit button on the right of the overview table. The edit Page opens up and all fields can be altered.

Date*
 

Amount*

Description*

Category*

Payment*

[Save Transaction](#)

Delete transaction

To delete a transaction, click on the  delete button on the right of the overview table.

Add Expense

Add Expense

To add an expense click on the “Add Expense” button. A new site is shown with the input fields to add a new Expense. Fill in all data and click the green “**Save Transaction**” button.

Input Fields:

Date: Date when the Transaction happened. Can also be in the past or future.

Amount: How much the Transaction in Swiss Francs was.

Description: Here is space to describe the transaction. E.g. Grocery Shopping Aldi

Category: Here you can select a category like e.g. Vacation, Shopping...

Payment: There are 5 payment types available - Visa, Mastercard, Bank, Cash, Crypto

Date*
 

Amount*

Description*

Category*
 

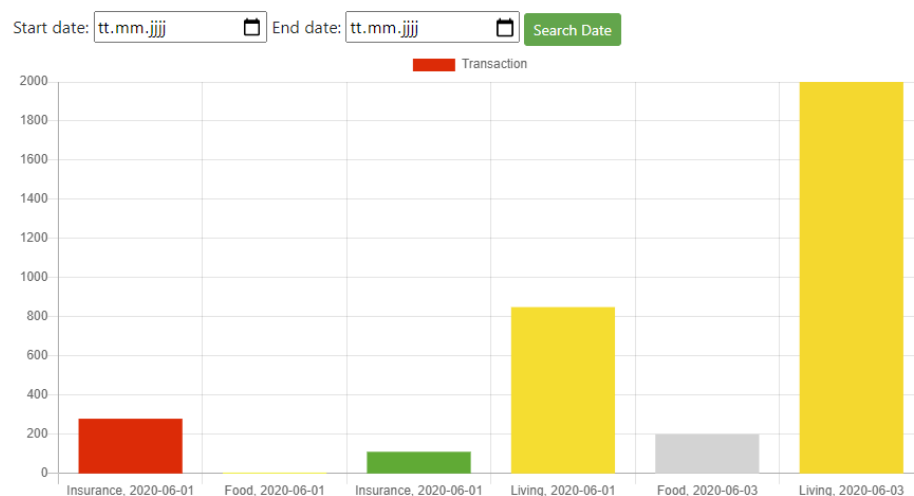
Payment*
 

Save Transaction

Show Bar chart

Show Barchart

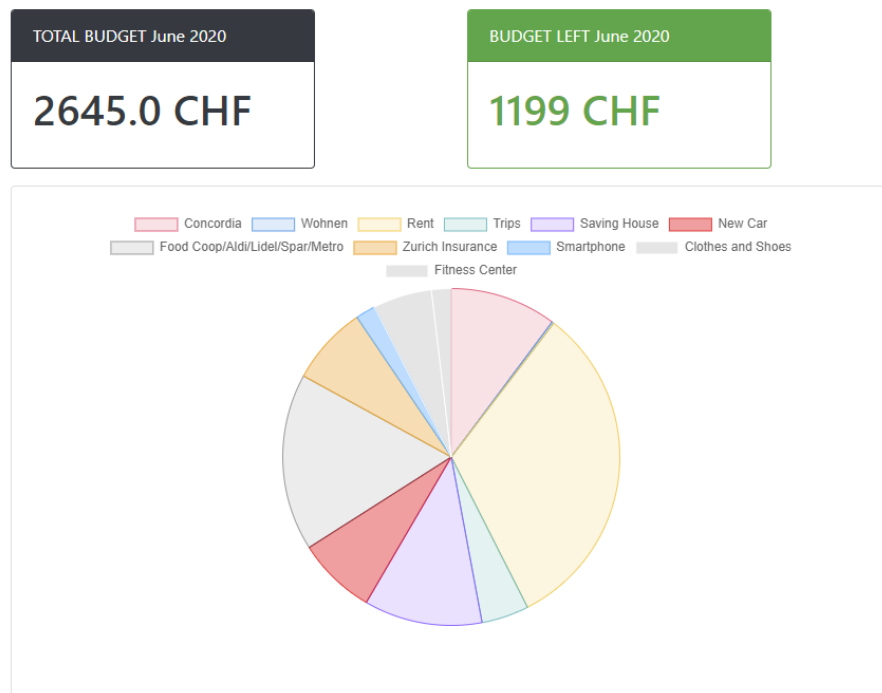
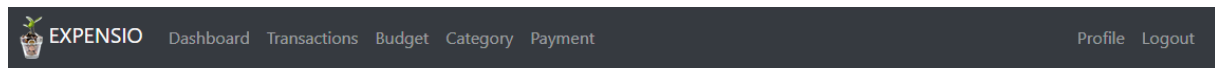
The “Show Bar chart” function, opens a page with a diagram where you can select the Date Range and all Transactions in this time period are shown.



Budget

The main page of the budget App shows all budget and an overview with charts and info cards of the current month.

Budgets show the different planned budgets which are available every month and all the budgets summed up are the total budget for the current month.



Budget for June 2020

[Add Budget](#)


Name	Amount	Description	Category	Date	Creator	Edit	Delete
Clothes and Shoes	150.00 CHF	New Clothes and Shoes	Clothes & Shoes	2020-06-03	kaoru		
Concordia	270.00 CHF	Health Insurance	Insurance	2020-05-31	kaoru		
Fitness Center	50.00 CHF	Fitness Center Schaan	Sport & Fun	2020-06-03	kaoru		



On top is the total budget for this month visible and the budget which is left for this month. The green color signals the user, that there is still budget left for expenses.

Should the budget be exhausted, the color changes to red, this indicates that the expenditure should be limited.

Edit budget

To edit a budget, click on the  edit button on the right of the overview table. The edit Page opens up and all fields can be altered.

Name*

Amount*

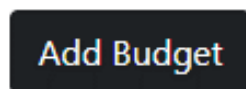
Description*

Category*

Delete budget

To delete a budget, click on the  delete button on the right of the overview table.

Add Budget



To add a budget click on the “Add Expense” button. A new site is shown with the input fields to add a new Expense. Fill in all data and click the green **“Save Budget”** button.

Input Fields:

Name: Name of the budget

Amount: How much the budget is in Swiss Francs.

Description: Here is space to describe the budget. E.g. Rent

Category: Here you can select a category like e.g. Vacation, Shopping...

Name*

Amount*

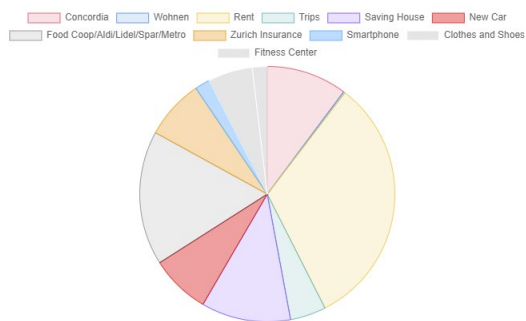
Description*

Category*

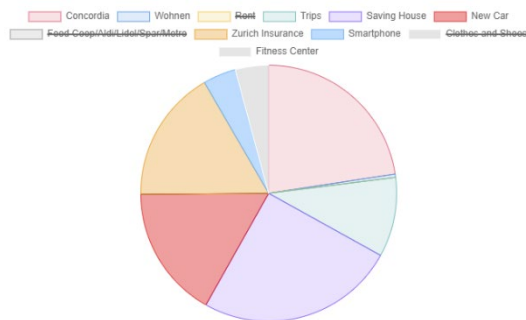
Pie Chart

The Pie Chart shows all budgets for the current month. With a click on the names of a budget, the clicked budget is no longer shown in the chart. Like that it is possible to get a better overview over expenses it you can deselect big expenses like rent.

ALL EXPENSES



EXLUDED BIG EXPENSES



Category

The main site shows all available categories.

Categories are used to group transactions together. Each transaction has a category. Possible categories are: Food, Clothes/Shoes, Housing, Vacations, Insurance. Categories have a creation date and a creator. You can edit or delete them with the two familiar icons on the right side.


[Dashboard](#)
[Transactions](#)
[Budget](#)
[Category](#)
[Payment](#)

[Profile](#)
[Logout](#)

Add Category

Name	Date	Creator	Edit	Delete
Clothes & Shoes	2020-06-03	kaoru		
Food	2020-06-01	kaoru		
Insurance	2020-05-31	kaoru		
Living	2020-06-01	kaoru		
Saving	2020-06-03	kaoru		
Sport & Fun	2020-06-03	kaoru		
Subscriptions	2020-06-03	kaoru		
Technology	2020-06-03	kaoru		
Vacation	2020-06-03	kaoru		

Edit category

To edit a category, click on the  edit button on the right of the overview table. The edit Page opens up and all fields can be altered.

Name*

Delete transaction

To delete a category, click on the  delete button on the right of the overview table.

Add Category

Add Category

To add a category click on the “Add Category” button. A new site is shown with the input fields to add a new Category. Fill in all data and click the green **“Save Category”** button.


Input Fields:

Name: Name of the Category like - Food, Clothes/Shoes, Housing, Vacations, Insurance










Name*

Payment


There are 5 basic types of payments to choose from: Visa, Bank, Mastercard, Crypto and Cash. It is possible to create more than one payment option like several Visa cards e.g. company and private card or a child bank account and a family bank account.


EXPENSIO
Dashboard Transactions Budget Category Payment
Profile Logout

Add Payment method

Card type	Description	Date	Edit	Delete
	from kaoru	2020-06-02		
	from alice	2020-05-13		
	from gerhard	2020-06-02		

Edit payment option

To edit a payment option, click on the  edit button on the right of the overview table. The edit Page opens up and all fields can be altered.

Type*

Description*

Date*

Delete payment option

To delete a payment option, click on the  delete button on the right of the overview table.

Add payment option

To add a payment option click on the “Add Payment” button. A new site is shown with the input fields. Fill in all data and click the green “Save Payment” button.

Input Fields:

Type: Visa, Bank, Mastercard, Crypto and Cash

Description: Name of the Category like - Food, Clothes/Shoes, Housing, Vacations, Insurance

Date: date of first usage.

Type*


Description*

Date*

User Profile

In User Profiles the user has access to the export and import function in addition to the user specifications.

Here it is possible to export all transactions, budgets, payments, or categories to a CSV file. The Upload button also provides an import function.

 EXPENSIO Dashboard Transactions Budget Category Payment Profile Logout

USER PROFILE

Hello, kaoru!

Your E-Mail address is: kaoru@mail.com

You are an important user since: May 31, 2020, 8:14 a.m.

Your last login was on: June 4, 2020, 8:40 a.m.

EXPORT YOUR DATA

Export transactionsExport budgetExport payment

UPLOAD CSV DATA

You can upload your transactions into your account from a csv file.
Your data has to look exactly like this. If you are unsure, please download an export first and edit the file.

Date	Amount	User	Payment	Description	Category
2020-06-01 00:00:00+00:00	80.00	user name	"Type: MASTERCARD, Description: from gerhard"	Coop	Food

Date: has to look like a DateTimeField from Django

Amount: this is the Transaction amount

User: this field will be ignored. Instead your currently logged in User will be used.

Payment: has be "Type: MASTERCARD, Description: text". You can choose between the options which are available under Payment.






Description: this is the Transaction description











Category: this category has to exist under Category

Datei auswählenKeine ausgewähltUpload file

EXPENSIO ©2020

Example of a csv export of all transactions:

Automatisches Speichern      transaction

Datei Start Einfügen Zeichnen Seitenlayout Formeln Daten Überprüfen Ansicht Hilfe A									
<div> <div>Einfügen</div> <div>  Ausschneiden  Kopieren  Format übertragen </div> </div> <div> <div>Zwischenablage</div> <div> <div>Calibri 11 A[^] A^v</div> <div> F <i>K</i> <u>U</u> </div> <div>  </div> <div>  </div> <div>  </div> </div> <div> <div>Schriftart</div> <div> <div>≡ ≡ ≡</div> <div>    </div> </div> <div> <div>Ausrichtung</div> <div> <div>ab Textumbruch</div> <div>  Verbinden und zentrieren </div> </div> </div> </div> </div>									

A1

✕ ✓ f_x

Date,Amount,User,Payment,Description,Category

Upload CSV file into EXPENSIO

EXPENSIO provides the functionality to upload a csv file to import transactions. A import file has to be a csv file and meet some requirements.

UPLOAD CSV DATA

You can upload your transactions into your account from a csv file.

Your data has to look exactly like this. If you are unsure, please download an export first and edit the file.

Date	Amount	User	Payment	Description	Category
2020-06-01 00:00:00+00:00	80.00	user name	"Type: MASTERCARD, Description: from gerhard"	Coop	Food

Date: has to look like a DateTimeField from Django

Amount: this is the Transaction amount

User: this field will be ignored. Instead your currently logged in User will be used.

Payment: has to be "Type: MASTERCARD, Description: text". You can choose between the options which are available under Payment.

Description: this is the Transaction description

Category: this category has to exist under Category

Datei auswählen

Keine ausgewählt

Upload file

Requirements:

- Date: must look like a DateTimeField from Django
- Amount: this is the Transaction amount
- User: this field will be ignored. Instead your currently logged in User will be used.
- Payment: has be of "Type: MASTERCARD, Description: text". You can choose between the options which are available under Payment.
- Description: this is the Transaction description
- Category: this category must exist under Category