

MyCash

sYSTEM dOCUMENTATION

07/06/2018

Content

[REQUIREMENTS 4](#_Toc517946085)

[A. User Requirements: 4](#_Toc517946086)

[B. System Requirements 5](#_Toc517946087)

[WORK PLAN 8](#_Toc517946088)

[A. First Season 8](#_Toc517946089)

[B. Second Season 8](#_Toc517946090)

[C. Third Season 8](#_Toc517946091)

[D. Fourth Season 9](#_Toc517946092)

[ARCHITECTURE 10](#_Toc517946093)

[A. Logical Architecture (Component Diagram) 10](#_Toc517946094)

[B. Physical Architecture (Deployment Diagram): 11](#_Toc517946095)

[DIAGRAMS 12](#_Toc517946096)

[A. CASE USE DIAGRAM: 12](#_Toc517946097)

[B. CASE USE DETAILS: 13](#_Toc517946098)

[Sign In (UC-SI): 13](#_Toc517946099)

[Sign Up (UC-SU): 13](#_Toc517946100)

[Modify User Profile (UC-MUP): 14](#_Toc517946101)

[See Historical Data (UC-SHD) 14](#_Toc517946102)

[See Technical Request (UC-SeTR): 15](#_Toc517946103)

[Send Technical Request (UC-SdTR): 15](#_Toc517946104)

[Search User (UC-SU): 15](#_Toc517946105)

[Delete User Account (UC-DUA): 16](#_Toc517946106)

[Delete Account (UC-DA): 16](#_Toc517946107)

[Add Category (UC-MC\_AC): 17](#_Toc517946108)

[Delete Category (UC-MC\_DC): 17](#_Toc517946109)

[Modify Category (UC-MC\_MC): 18](#_Toc517946110)

[Manage Incomes/Expenses Summary (UC-MIES): 18](#_Toc517946111)

[Log Out (UC-LO): 19](#_Toc517946112)

[Recover Account (UC-RA): 19](#_Toc517946113)

[Personalize (UC-P): 19](#_Toc517946114)

[Add Incomes (UC–MI\_AI): 20](#_Toc517946115)

[Delete Incomes (UC-MI\_DI): 20](#_Toc517946116)

[Modify Incomes (UC-MI\_MI): 20](#_Toc517946117)

[Add Expenses (UC-ME\_AE): 21](#_Toc517946118)

[Delete Expenses (UC-ME\_DE): 21](#_Toc517946119)

[Modify Expenses (UC-ME\_ME): 21](#_Toc517946120)

[Manage Goals (UC-MG): 22](#_Toc517946121)

[C. SEQUENCE DIAGRAMS: 23](#_Toc517946122)

[Sign In (SD-SI): 23](#_Toc517946123)

[Sign Up (SD-SU): 24](#_Toc517946124)

[Modify User Profile (SD-MUP): 24](#_Toc517946125)

[See Historical Data (SD-SHD): 25](#_Toc517946126)

[See Technical Request (SD-SeTR) 25](#_Toc517946127)

[Send Technical Request (SD-SdTR) 26](#_Toc517946128)

[Search User (SU): 26](#_Toc517946129)

[Delete User Account (SD-DUA): 27](#_Toc517946130)

[Delete Account (SD-DA): 27](#_Toc517946131)

[Add Category (SD-MC\_AC): 28](#_Toc517946132)

[Delete Category (SD-MC\_DC): 28](#_Toc517946133)

[Modify Category (SD-MC\_MC): 29](#_Toc517946134)

[29](#_Toc517946135)

[Manage Incomes/Expenses Summary (SD-MIES): 30](#_Toc517946136)

[Log Out (SD-LO): 31](#_Toc517946137)

[Recover Account (SD-RA): 31](#_Toc517946138)

[Personalize (SD-P): 32](#_Toc517946139)

[Add Income (SD-MI\_AI): 32](#_Toc517946140)

[Delete Income (SD-MI\_DI): 33](#_Toc517946141)

[Modify Income (SD-MI\_MI): 33](#_Toc517946142)

[Add Expense (SD-ME\_AE): 34](#_Toc517946143)

[Delete Expense (SD-ME\_DE): 34](#_Toc517946144)

[Modify Expense (SD-ME\_ME): 35](#_Toc517946145)

[Add Goal (SD-MG\_AG): 36](#_Toc517946146)

[Delete Goal (SD-MG\_DG): 36](#_Toc517946147)

[RELATIONAL DIAGRAM: 38](#_Toc517946148)

[CLASS DIAGRAM 39](#_Toc517946149)

[EXTRA DOCUMENTATION 40](#_Toc517946150)

[A. Meetings: 40](#_Toc517946151)

[1. Meeting Agenda [M3\_01] 40](#_Toc517946152)

[2. Meeting Agenda [M3\_02E] 41](#_Toc517946153)

[3. Meeting Agenda [M3\_03] 42](#_Toc517946154)

[4. Meeting Agenda [M3\_03] 43](#_Toc517946155)

# 

# REQUIREMENTS

## User Requirements:

|  |  |  |
| --- | --- | --- |
| Id User Requirement [UR] | Requirement | Class |
| UR-1-SI B  Sign In | The user can log in by entering his/her main email and password. | Functional |
| UR-2-SU A  Sign Up | The user can register in the system, entering his/her name, email, password, phone (optional). | Functional |
| UR-3-MUP C  Modify User Profile | The user can modify in their profile their registration information (names, email, password, etc.). | Not Functional |
| UR-4-SHD G  See Historical Data | You can see a history of income and expenses. | Functional |
| UR-5-SeTR K  See Technical Request | The Administrator can see Technical Requests. | Not Functional |
| UR-6-SdTR J  Send Technical Request | The user can send technical requests to the Administrator to help improve the system. | Not Functional |
| UR-7-SU L  Search User | The Administrator can see a list of users. | Not Functional |
| UR-8-DUA M  Delete User Account | The Administrator can delete a selected user. | Functional |
| UR-9-DA H  Delete Account | The user can delete his account. | Functional |
| UR-[10-12]-MC I  Manage Categories | The user can add/edit/delete categories of incomes or expenses. | Not Functional |
| UR-13-MIES F  Manage Income/Expenses Summary | A summary of the information on income, expenses and savings / losses will be displayed. | Functional |
| UR-[15-17]-MI D  Manage Incomes | The user can manage their income. | Functional |
| UR-[18-20]-ME E  Manage Expenses | The user can manage their expenses. | Functional |
| N GOALS |  |  |
|  |  |  |

## System Requirements

|  |  |  |
| --- | --- | --- |
| Id System Requirement  [SR] | Requirement | Class |
| SR-2-SU A  Sign Up | There will be a form for the user to register in the system, where it will be necessary to enter the name, main email, phone (optional) and the password twice, for security reasons, and it will also be validated with regular expressions that the entry fields are filled properly (e.g.: name must contain only letters). | Functional |
| SR-2-SU A  Sign Up | The registration information will be stored in a relational database (PostgreSQL). | Functional |
| SR-1-SI B  Sign In | There will be a form to login (main email and password), and regular expressions will be used to verify that the email is properly written. | Functional |
| UR-3-MUP C  Modify User Profile | Changes made to user information will be made in the database. | Not Functional |
| UR-3-MUP C  Modify User Profile | When modifying a field of the user information, it will be verified with regular expressions that the new value entered has the appropriate form. | Not Functional |
| UR-[15-17]-MI D  Manage Incomes | The added income will be stored in the database, with the following information: name of the income, amount, category (work, rents, sales, etc.), effective date and creation date. | Functional |
| UR-[10-12]-MC I  Manage Categories | A list will be presented with the category of the income, and within each category there will be a list of income, you can select an income and see all your detailed information, edit all your respective information, or eliminate the income. | Functional |
| UR-[10-12]-MC I  Manage Categories | A new income classification may be added. | No Functional |

|  |  |  |
| --- | --- | --- |
| UR-[18-20]-ME E  Manage Expenses | The added expenses will be stored in the database, with the following information: name of the expense, amount, category (services, loans, food, etc.), effective date and creation date. | Functional |
| 11 | A list will be presented with the category of the expenses, and within each category there will be a list of expenses, you can select an expense and see all your detailed information, edit all your respective information, or eliminate the expense. | Functional |
| 12 | A new classification of expenses may be added. | Not Functional |
| 14 | Several graphs will be displayed to visualize the summary information of income, expenses and savings / losses (bar chart, double bar chart, pie chart and calendars), for each month and at day intervals. | Functional |
| 15 | Additional information about income and expenses will be displayed, with numeric data. For example: "Year: 2015, Month: May, Income: + $ 5000.00". | Functional |
| 16 | Information on savings / loss in numerical form corresponding to a period of time will be displayed. For example:    - "Year: 2016, Month: February, Savings: + $ 4895.00".    - "Year: 2017, Month: August, Day: 14, Loss: - $ 240.00". | Functional |
| 17 | When selecting a day you can see a list of the income or expenses of that day, separated by their classifications. | Functional |
| 18 | A calendar will be displayed so that you can select a specific day, then you will be able to see the expenses and incomes of that day, and edit them. | Functional |
| 19 | The days of the calendar will be painted with colors, red or green, red when that day money has been lost, and the tonality will be redder when the loss is greater and will be more transparent when the loss is less; and green when there has been a saving, and the tonality will be greener when the saving is greater and more transparent when the saving is lower. | Not Functional |
| 20 | If you delete your account, the account will go into an inactive state within the database. If the user tries to enter his inactive account, he can recover it. | Not Functional |

|  |  |  |  |
| --- | --- | --- | --- |
| 21 | I | The user can add categories (consisting only of the name of the category) to incomes or expenses, delete existing categories or edit a category (change its name). | Not Functional |
| 22 | J | The user can send Technical Requests to the Administrator so that he/she can manage them and improve the system.  The Technical Request consist of message and the ID of the user. | Not Functional |
| 23 | K | The Administrator can see a list of Technical Requests and set them as Checked so that the Technical Request don’t appear again in his list. | Not Functional |
| 24 | L | The Administrator can see a list of the users with their basic information (name, email and state) and also can select a user of the list and delete him/her (change his/her state to inactive). | Not Functional |

# WORK PLAN

## First Season

## Second Season

## Third Season

**Roles:**

Percy Maldonado Quispe Designer percy.maldonado@ucsp.edu.pe

Alonso Jesús Cerpa Salas Programmer alonso.cerpa@ucsp.edu.pe

Jesamin Melissa Zevallos\* Analyst jesamin.zevallos@ucsp.edu.pe

José David Mamani Vilca Project Manager jose.mamani.vilca@ucsp.edu.pe

Guillermo Enrique Calderón Quispe Stakeholder gcalderon@ucsp.edu.pe

**Tasks: Dates: Roles:**

1. Presentation 2: 10/05/18 – 10/05/18 Full Team and Stakeholder
2. Meeting M3\_01: 15/05/18 – 15/05/18 Full Team
3. Emergency Meeting M3\_02: 17/05/18 – 17/05/18 Full Team
4. Test Cases: 10/05/18 – 05/06/18 Alonso, Jesamin
5. Final Architecture: 14/05/18 – 24/05/18

5.1 Physical Architecture: 14/05/18 – 17/05/18 Percy

5.2 Logical Architecture: 18/05/18 – 24/05/18 Percy

6. Exhaustive Meeting M3\_04: 24/05/18 – 24/05/18 Full Team

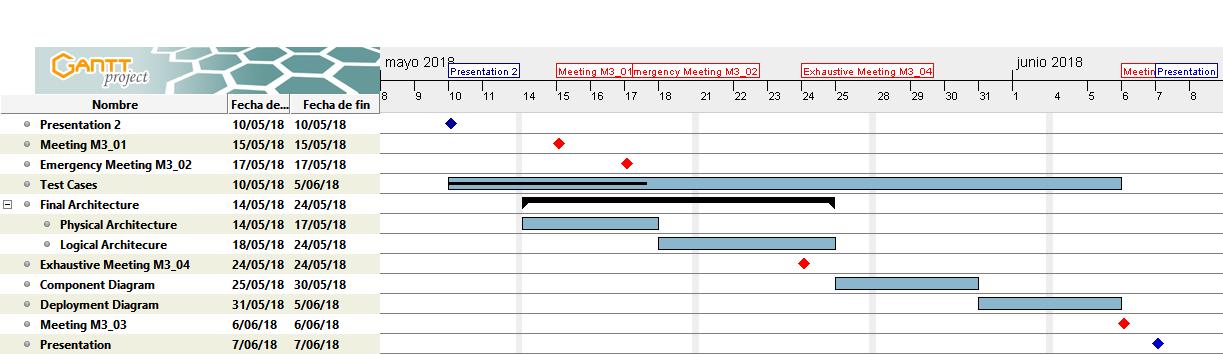
7. Component Diagram: 25/05/18 – 30/05/18 Percy

8. Deployment Diagram: 31/05/18 – 05/06/18 Percy

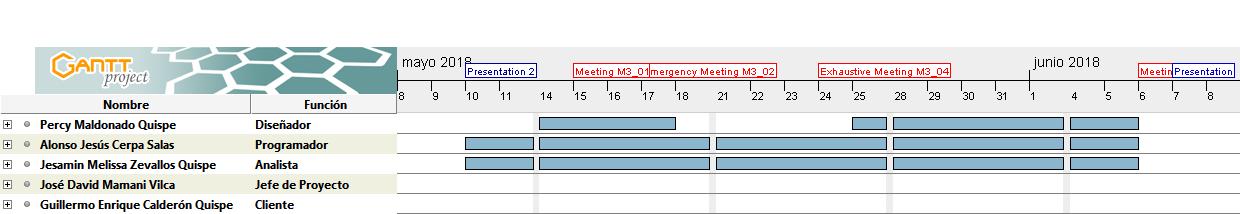
9. Meeting M3\_03: 06/06/18 – 06/06/18 Full Team

10 Presentation: 07/06/18 – 07/06/18 Full Team and Stakeholder

**Gantt Task Diagram:**



**Gantt Roles Diagram**



## Fourth Season

**Roles:**

Percy Maldonado Quispe Designer percy.maldonado@ucsp.edu.pe

Alonso Jesús Cerpa Salas Programmer alonso.cerpa@ucsp.edu.pe

Jesamin Melissa Zevallos Analyst jesamin.zevallos@ucsp.edu.pe

José David Mamani Vilca Project Manager jose.mamani.vilca@ucsp.edu.pe

Guillermo Enrique Calderón Quispe Stakeholder gcalderon@ucsp.edu.pe

**Tasks: Dates: Roles:**

1. Presentation 3: 07/06/18 – 07/06/18 Full Team and Stakeholder
2. Meeting M4\_01: 08/06/18 – 08/06/18 Full Team
3. Deployment Plan: 08/06/18 – 19/06/18 Percy, David
4. Missing Use Cases: 08/06/18 – 19/06/18 Alonso
5. Meeting M4\_02 20/06/18 – 20/06/18 Full Team

6. Installer: 20/06/18 – 29/06/18 Alonso

7. Report of Test Cases: 20/06/18 – 29/06/18 Jesamin

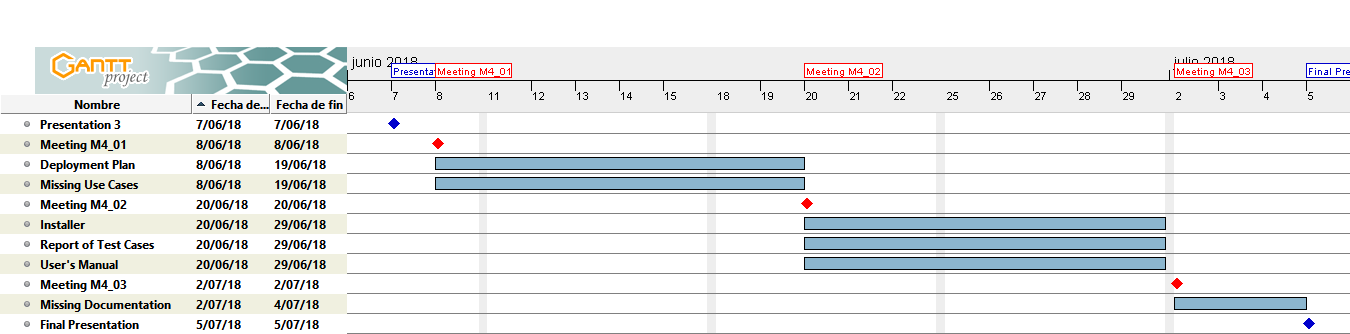
8. User’s Manual 20/06/18 – 29/06/18 Percy, David

9. Meeting M4\_03: 02/07/18 – 02/07/18 Full Team

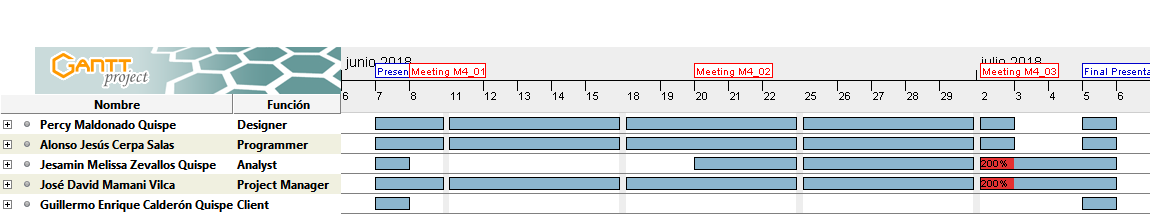
10 Missing Documentation: 02/07/18 – 04/07/18 Jesamin, David

11. Final Presentation: 05/07/18 – 05/07/18 Full Team and Stakeholder

**Gantt Task Diagram:**



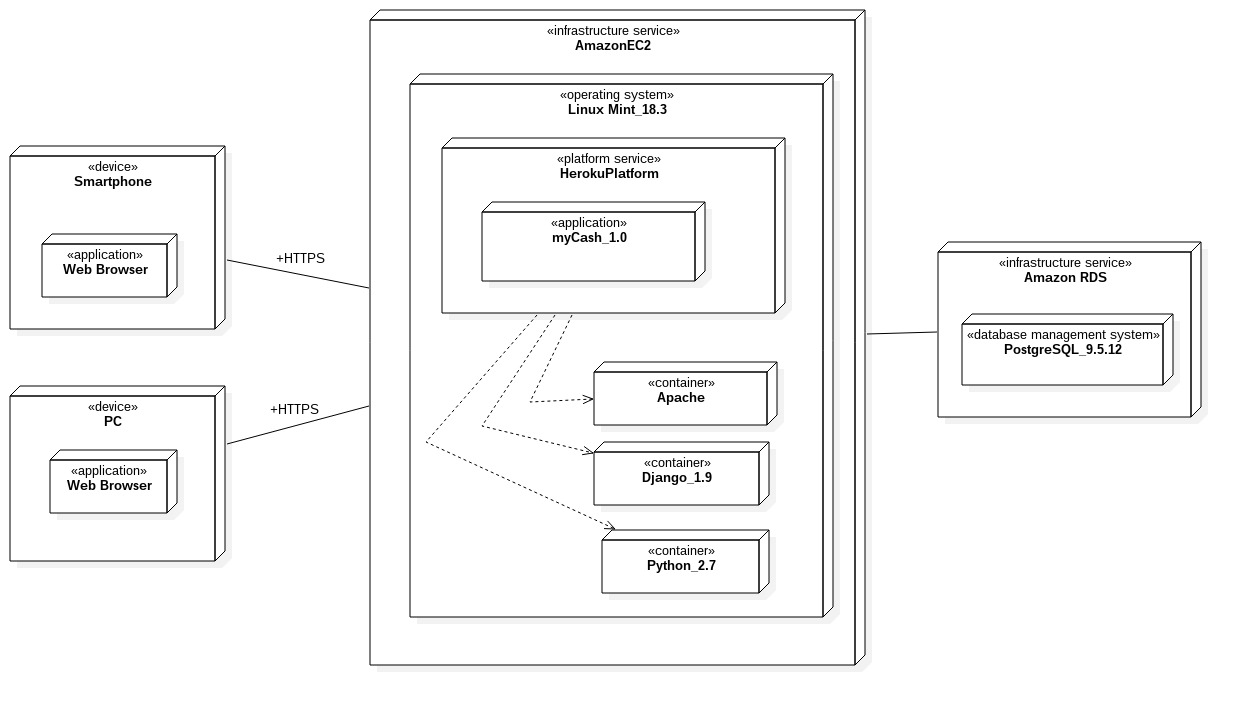
**Gantt Roles Diagram:**



# ARCHITECTURE

# C:\Documents and Settings\Administrator\Desktop\Image2.EMFLogical Architecture (Component Diagram)

# Physical Architecture (Deployment Diagram):



# DIAGRAMS

## CASE USE DIAGRAM:

C:\Documents and Settings\Administrator\Desktop\Image2.EMF

## CASE USE DETAILS:

### Sign In (UC-SI):

**Code:** UC-SI

**Objective:**

    Allows Users and Administrators to get into their account. Also, makes a differentiation between roles.

**Pre-requisites:**

    1: For a complete functionality in this use case, the user must exist in the database.

    2: This use case won´t be available if you are already logged in.

**Main flow:**

    1: A form called "Login Form" is displayed with two fields: Email and Password.

    2: The system user types his/her email and password.

    3: The system validates incoming information.

    4: The system designs a request and send it to the database.

    5: The database returns a Boolean answer with value set as "True".

    6: The account can be accessed.

**Secondary flow:**

    3.1: If data validation fails, an error message is shown next to a possible solution for the user. The "Login Form" won't be closed.

    5.1: If the database returns an answer set as "False", then an error message is shown next to the form. The "Login Form" won't be closed.

### Sign Up (UC-SU):

**Code**: UC-SU

**Objective:**

    Registers and saves data from new users.

**Pre-requisites:**

    1: There is not a specific pre-requisite to apply this use case.

**Main Flow:**

    1: A form called "Sign Up form" is displayed to acquire necessary information.

    2: The user types his/her personal information: Name, Email, Password, Extra Email and Phone Number.

    3: The user clicks "Create account" button.

    4: The system validates incoming data.

    5: Once validation process has ended, the system designs a query to the database.

    6: The database searches for accounts with the same email.

    7: Previous operation will return a Boolean answer. If the returned value is equal to "True", then new data is accepted.

    8: The system displays a success message. Subsequently, the user is redirected to the profile section.

**Secondary Flow:**

    4.1 If data validation fails, then an error message is shown.

    7.1 In case of a false answer, incoming data is not accepted and the user must write a new account with different information.

### Modify User Profile (UC-MUP):

**Code**: UC-MUP

**Objective:**

Allows any kind of user to modify his/her personal data.

**Pre-requisites:**

    1: The user must be logged in the application.

    2: The user must be in the "Modify your profile" section.

**Main Flow:**

    1: A form is displayed to receive new information. There are five fields to complete: Name, Email, Password, Extra email and Phone number.

    2: The user types his/her new personal information. It's not necessary to complete all the fields.

    3: The system validates incoming data.

    4: Once validation process has ended, the system design a query for the database.

    6: The database looks for accounts with the same email.

    7: Previous operation will return a Boolean answer. If the returned value is equal to "True", then new data is accepted.

    8: A success message is displayed. Now, the user can see a new profile with different information.

**Secondary Flow:**

    7.1 In case of a false answer, a message error is displayed. Most of the time this problem could be related with the email section, so this possible solution must be shown to the user.

### See Historical Data (UC-SHD)

**Code:** UC-SHD

**Objective:**

Makes an exhaustive analysis from the last weeks, months and even years.

**Pre-requisites:**

    1: The user must be logged in the application.

    2: The daily balance must be pre-worked each time an expense or income is added or deleted.

**Main Flow:**

     1: The user accesses to the historical data section.

     2: Once clicked the button for this section, the system proceeds to design a query to the database.

     3: The database returns a list with the information of the n- last months. (This n is defined by the system).

     4: The system organizes incoming information from the database. A list of n-last months (This n is defined by the user) is organized with the balance from different days. Each day will have a color that defines the daily balance. A green color indicates a positive balance. A red color indicates a bad day.

     5: The system shows a structured list from the last n-months.

    4: If a user selects a specific day, a modal appears and shows every incomes and expenses in that day.

### See Technical Request (UC-SeTR):

**Code:** UC-SeTR

**Objective:**

    The request pool is a place where all technical request are stored. There is just one request pool and only administrators will have the access to it.

**Pre-requisites:**

    1: The administrator must logged in the system.

**Main Flow:**

    1: The administrator accesses to the request pool section.

    2: The administrator reviews the last n-technical request stored in the pool. Oldest request will be at the top of the list.

    3: The administrator selects a specific request to be solved.

    4: If a solution is found for any technical request, then this is marked as "solved" by the administrator.

    5: The system adds a solution date and writes administrator's ID next to the request.

    6: The request is now saved in the database as a solved issue.

### Send Technical Request (UC-SdTR):

**Code:** UC-SdTR

**Objective:**

    Sends a request to the administrator in case of any problem.

Pre-requisites:

    1: The user must be logged in the system.

**Main Flow:**

    1: The user accesses to the "Report a Fail" section.

    2: The user types the description of the fault (Required).

    3: The user selects an option:

        3.1: Send Solicitude

        3.2: Cancel

    4: In case of "Send Solicitude", the system adds user information and designs a query to send the technical request to the pool. Otherwise the "Report a Fail" section is closed.

       4.1: The technical request is stored as an unsolved issue.

    5: Once done, the section is closed. Subsequently a success message is shown.

### Search User (UC-SU):

**Code:** UC-SU

**Objective:**

    Searches a user.

**Pre-requisites:**

    1: The administrator must logged in the system.

**Main Flow:**

    1:  A form is displayed with different fields to make a better search.

    2: The administrator types the information in the fields.

    3: The system accesses to the database.

    4: The system returns a list of founded records.

    5: The list is shown from newest users to oldest.

### Delete User Account (UC-DUA):

**Code:** CU-DUA

**Objective:**

    Modify the status of a user.

**Pre-requisites:**

    1: The administrator must logged in the system.

**Main flow:**

    1: The administrator enters the panel to delete accounts.

    2: Search a User.

    3: A text box is displayed.

    4: The administrator types the reason why the user will be banned.

    5: The administrator modifies the status of the account.

    6: Select: Save or Cancel.

    7: If "save" is selected, the system change the visibility of the account. Now the user cannot access to his/her account even if he/she uses the recovery function.

**Secondary Flow:**

     7.1 If "cancel" button is selected all changes will be ignored and the section will be closed.

### Delete Account (UC-DA):

**Code:** UC-DA

**Objective:**

    Inactivates your account.

**Pre-requisites:**

    The system user must be logged in the system.

**Main flow:**

    1: Inside the profile page you click the button "Delete account".

    2: Appears a form that asks you to enter your password in order to delete your account.

    3: If the password was correct, inside the database your account now has a type: inactive. All the payment plans will be canceled and the system automatically logs out and you are redirected to the login page.

**Secondary Flow:**

         3.1: Otherwise, it shows you a message "Invalid password, enter it again".

### Add Category (UC-MC\_AC):

**Code:** UC-MC\_AC

**Objective:**

    Allow users to create a new category.

**Main flow:**

    1: The user presses "Add to my categories" button located next to the new income or expense.

    2: A dropdown menu is displayed with all user's categories. The user selects the last option called "Create a new category".

    3: A form called "Create Category" is displayed.

    4: The user types a new category name.

    5: The user clicks "Save" button.

    6: The system designs a request to find out another category with the same name.

    7: If the system returns a true answer, then the new category is created and the current income or expense is added to it.

**Secondary flow:**

    1.1: If the user do not want to assign a category for the current expense or income, then a category called "Others" will save the new expense or income.

    1.2: You can create a new category from the category section. The creating process is the same with just one differences: The new category will be empty at the beginning.

7.1: In case of a false answer, an error message is shown. This message will contain a possible solution to this problem.

### Delete Category (UC-MC\_DC):

**Code:** UC-MC\_DC

**Objective:**

    Allows user to delete an existing category. There some non-obligatory conditions that should be taken in account.

**Local Pre-requisites:**

    1:  The user must be in the category section.

**Main flow:**

    1: Each category listed in this section will have a "Delete" button next to it. To delete a specific category, the user must clicks the "Delete" button.

    2: A confirmation message is shown with an unmarked field.

    3: The user confirms the action selecting the unmarked box.

    4: The system proceeds to delete the selected category. If the category is not empty, the system will send the resident elements to the default category ("Others").

    5: Once deleting process is ended, a success message is shown.

**Secondary flow:**

   1.1 The default category "Others" cannot be deleted. For this, the delete button won't appear next to it.

   5.1 If deleting process fails, an error message is displayed.

### Modify Category (UC-MC\_MC):

**Code:** UC-MC\_MC

**Objective:**

    Allows users to change the name or administrate resident elements in a category. Some restrictions are defined in secondary flow.

**Local Pre-requisites:**

    1: The user must be in the category section.

**Main flow:**

    1: The user clicks "Modify" button in a specific category.

    2: A form is display with three fields: A name, a list of elements contained in that category and a dropdown menu. Each element can be selected in order to be added in another category. The third field contained a list of categories at which the selected elements will be sent.

    3: The user types a new name (not necessary) or selects the elements to save them in a new category. By default, the dropdown menu will start with "Others" category.

    4: The user clicks "Save" button.

    5: The system will update the name of the category previous validation. (And only if it's necessary).

    6: The system will remove the selected elements from the current category. Removed elements will be now stored in another category.

    7: A success message is shown.

**Secondary flow:**

* 1. The default category "Others" cannot be modify. Resident elements are exonerated from this privilege.

5.1 If the validation fails, coming operations will be canceled. An error message is shown with some possible solutions for the user.

### Manage Incomes/Expenses Summary (UC-MIES):

**Code**: UC-MIES

**Objective:**

    Shows an organized diagram with user's information.

**Pre-requisites:**

    1: The system user must be logged in the application.

**Main flow:**

    1: The user accesses to the overview section.

    2: User's information is obtained [income / expenses - Objectives].

    3: Defined goals and their progress are shown in a segment. This section is called "Goals".

    4: User's information is organized a displayed in two ways:

        4.1: Frequency: Monthly.

        4.2: Type of Graph: Linear.

    5: Also, the user can:

        5.1 Change the type of Graphic.

            - Bars.

            - Linear

            - Circular.

            - Area.

            - Histogram.

        5.2 Change the Frequency.

            - Weekly

            - Monthly.

            - Annual.

### Log Out (UC-LO):

**Code:** UC-LO

**Objective:**

Takes you out of session and the next time you want enter to the system it will ask you for your password.

**Pre-requisites:**

    1: The system user must be logged in the application.

**Main Flow:**

    1: If you click in your profile section, a modal will appear with summed information of your profile and also a "Log out" button.

    2: If you click in the "Log out" button, you will be redirected to the login page and the next time you want to enter to the system it will ask you for your password.

### Recover Account (UC-RA):

**Code:** UC-RA

**Objective:**

    Makes the type of your account active so that you can use it again.

**Pre-requisites:**

    There are not specific requisites to apply this use case.

**Main Flow:**

1: You try to login as a user that deleted his/her account.

2. If the data entered was correct.

     2.1 An email is sent with a link so you can login properly.

     2.2: You will be redirected to a page that says that you are recovering your account and you must go to your respective email to confirm the recovering.

     2.3 If you click on the link, the database will change the visibility of your account to active, and you will go to the overview page of your account as a normal login.

### Personalize (UC-P):

**Code:** UC-P

**Objective:**

Change the colors and the fonts of the design of the system.

**Pre-requisites:**

    1: The user must be logged in the system.

**Main Flow:**

    1: Inside your profile settings, you click on the button "Personalize".

    2: A box will be displayed. In this box a palette will allow you to change the system colors. Also, a list of fonts will allow you to change the system letters. Finally, at the end of the box, there will be 2 buttons "Accept" and "Cancel".

    3: If user clicks on "Accept" button, changes will be applied. Otherwise, the previous system aspect will be saved without changes.

### Add Incomes (UC–MI\_AI):

**Code:** UC-MI\_AI

**Pre-requisites:**

    1: The user must be logged in the application.

**Main Flow:**

    1: A form is display to acquired necessary information.

    2: The user must type in the next fields: Name [Obligatory], Amount [Obligatory], Category [Not obligatory] and Date [Automatic].

    3: Selects "Create" option.

    4: The system will validate incoming data.

    5: Once completed the validation process, new data will be sent to the database.

    6: A success message is displayed.

**Secondary Flow:**

    5.1 If validation fails, an error message is shown and all changes will be discard.

### Delete Incomes (UC-MI\_DI):

**Code:** UC-MI\_DI

**Pre-requisites:**

    1: The user must be logged in the application.

**Main Flow:**

    1: The user selects an existing income.

    2: The user presses the "delete" button.

    3: The user confirms the action.

    4: A success message is displayed.

**Secondary Flow:**

    3.1 Otherwise, all changes will be discard.

### Modify Incomes (UC-MI\_MI):

**Code:** UC-MI\_MI

**Objective:**

Allows user to modify a particular income. Some conditions will be required.

**Pre-requisites:**

    1: The user must be logged in the application.

**Main Flow:**

    .1: The user selects a particular income.

     2: A form called "Modify Income" is displayed in a modal. This form has 4 fields: Name [Obligatory], Amount [Obligatory], Category [Obligatory] and Date [Automatic].

     3: The user clicks "Save" button.

     4: The system will validate incoming data.

     5: Once completed the validation process, new data will be sent to the database.

     6: A success message is displayed.

**Secondary Flow:**

5.1 If validation fails, an error message is shown and all changes will be discard.

### Add Expenses (UC-ME\_AE):

**Code:** UC-ME\_AE

**Pre-requisites:**

    1: The user must be logged in the application.

**Main Flow:**

    1: A form is display to acquired necessary information.

    2: The user must type in the next fields: Name [Obligatory], Amount [Obligatory], Category [Not obligatory] and Date [Automatic].

    3: Selects "Create" option.

    4: The system will validate incoming data. This action will prevent some attacks in the system.

    5: Once completed the validation process, new data will be sent to the database.

    6: A success message is displayed.

**Secondary Flow:**

    3.1 Otherwise, all changes will be discard.

   5.1 If validation fails, an error message is shown and all changes will be discard.

### Delete Expenses (UC-ME\_DE):

**Code:** US-ME\_DE

**Pre-requisites:**

    1: The user must be logged in the application.

**Main Flow:**

    1: The user selects an existing expense.

    2: The user presses the "delete" button.

    3: The user confirms the action.

    4: A success message is displayed.

**Secondary Flow:**

    3.1 Otherwise, all changes will be discard.

### Modify Expenses (UC-ME\_ME):

**Code:** UC-ME\_ME

**Pre-conditions:**

The user must be logged in the application.

**Objective:**

Allows user to modify a particular income. Some conditions will be required.

**Pre-requisites:**

    1: The user must be logged in the application.

**Main Flow:**

    .1: The user selects a particular expense.

     2: A form called "Modify Expense" is displayed in a modal. This form has 4 fields: Name [Obligatory], Amount [Obligatory], Category [Obligatory] and Date [Automatic].

     3: The user clicks "Save" button.

     4: The system will validate incoming data.

     5: Once completed the validation process, new data will be sent to the database.

     6: A success message is displayed.

**Secondary Flow:**

      5.1 If validation fails, an error message is shown and all changes will be discard.

### Manage Goals (UC-MG):

**Code:** UC-MG

**Objective:**

Shows a list of objectives that indicates how much your goals are completed. This is a good way to indicate you what kind of things you can buy.

**Pre-requisites.**

    1: The user must be logged in the application.

**a) Add Goal:**

**Main Flow:**

        1: In the overview page exits a button called "Goals". If a user clicks in it, a modal will be deployed from the right side.

        2: The system will make a list from the user's goals. Also, this list includes a progress percentage.

        3: Inside this section, a button called "Add a Goal" allows user to create new goals.

        4: When a user clicks the "Add a goal" button, a form is display to acquire necessary information.

        5: There are four field to be completed: Name [Obligatory], Amount [Obligatory], Dead Point [Optional], Date of Creation [Automatic]. The system will add extra information: State and Progress.

        6: Once done, the user clicks "Save Goal" button.

        7: The system will save the new goal. Now your goal is added, and you will see it in the overview page, inside the list of goals.

**Secondary Flow:**

        6.1 If the user clicks the "Cancel" button, acquired information will be discard.

**b) Delete Goal**

    Local Pre-requisites:

        The user should be in Goals Section.

**Main Flow:**

        1. Inside Goals Section, a list of user goals is shown. Each goals has a delete button next to it.

        2. The user clicks "Delete" button.

        3. A confirmation message appears.

        4. The user confirms the action.

        5. The system receives the answer and sends a request to the database.

        6. Database returns a Boolean answer equal to "True", it means that the request was successfully completed.

        7. A success message is shown.

**Secondary Flow:**

        4.1 Otherwise confirmation message is closed without nothing been affected.

        6.1 In case of a negative answer, an error message is shown once the confirmation message was closed.

## SEQUENCE DIAGRAMS:

### Sign In (SD-SI):

C:\Documents and Settings\Administrator\Desktop\Image2.EMF

### C:\Documents and Settings\Administrator\Desktop\Image2.EMFSign Up (SD-SU):

### Modify User Profile (SD-MUP):

C:\Documents and Settings\Administrator\Desktop\Image2.EMF

### See Historical Data (SD-SHD):

C:\Documents and Settings\Administrator\Desktop\Image2.EMF

### See Technical Request (SD-SeTR)

C:\Documents and Settings\Administrator\Desktop\Image2.EMF

### Send Technical Request (SD-SdTR)

C:\Documents and Settings\Administrator\Desktop\Image2.EMF

### C:\Documents and Settings\Administrator\Desktop\Image2.EMFSearch User (SU):

### Delete User Account (SD-DUA):

C:\Documents and Settings\Administrator\Desktop\Image2.EMF

### Delete Account (SD-DA):

C:\Documents and Settings\Administrator\Desktop\Image2.EMF

### Add Category (SD-MC\_AC):

C:\Documents and Settings\Administrator\Desktop\Image2.EMF

### Delete Category (SD-MC\_DC):

C:\Documents and Settings\Administrator\Desktop\Image2.EMF

### Modify Category (SD-MC\_MC):

### C:\Documents and Settings\Administrator\Desktop\Image2.EMF

### Manage Incomes/Expenses Summary (SD-MIES):

C:\Documents and Settings\Administrator\Desktop\Image2.EMF

### Log Out (SD-LO):

C:\Documents and Settings\Administrator\Desktop\Image2.EMF

### Recover Account (SD-RA):

C:\Documents and Settings\Administrator\Desktop\Image2.EMF

### Personalize (SD-P):

C:\Documents and Settings\Administrator\Desktop\Image2.EMF

### Add Income (SD-MI\_AI):

C:\Documents and Settings\Administrator\Desktop\Image2.EMF

### Delete Income (SD-MI\_DI):

C:\Documents and Settings\Administrator\Desktop\Image2.EMF

### Modify Income (SD-MI\_MI):

C:\Documents and Settings\Administrator\Desktop\Image2.EMF

### C:\Documents and Settings\Administrator\Desktop\Image2.EMFAdd Expense (SD-ME\_AE):

### Delete Expense (SD-ME\_DE):

C:\Documents and Settings\Administrator\Desktop\Image2.EMF

### Modify Expense (SD-ME\_ME):

C:\Documents and Settings\Administrator\Desktop\Image2.EMF

### C:\Documents and Settings\Administrator\Desktop\Image2.EMFAdd Goal (SD-MG\_AG):

### Delete Goal (SD-MG\_DG):

C:\Documents and Settings\Administrator\Desktop\Image2.EMF

ENTITY RELATIONSHIP DIAGRAM:

C:\Documents and Settings\Administrator\Desktop\Image2.EMF

## RELATIONAL DIAGRAM:

C:\Documents and Settings\Administrator\Desktop\Image2.EMF

## C:\Documents and Settings\Administrator\Desktop\Image2.EMFCLASS DIAGRAM

# Test Cases

### Sign Up:

|  |  |
| --- | --- |
| Test Case | |
| Test ID: TC-SU01 | Designed by: MyCash |
| Priority: High | D. Date: 03-06-18 |
| Test Title: Sign Up | E. Date: 04-06-18 |
| Exec. By: Zevallos | |
| Classes Scope: SignUpView - UserManager - User | |
| Environment: MyCash - Computer - DjangoServer - PyCharm | |
| Dependencies: User does not have a created account | |
| Related Cases: LoginView – OverviewView | |
| Objective: Registers and saves data from new users | |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| N | Test Steps | Test Data | Expected Result | Actual Result | Status |
| 1 | Exists Email, Valid Password | “p[maldonado@gmail.com](mailto:maldonado@gmail.com)”,  ”maldonado” | Not Sign Up | Not Sign Up | Pass |
| 2 | Exists Email, Invalid Password | “[juan@gmail.com](mailto:juan@gmail.com)”,  “juan” | Not Sign Up | Not Sign Up | Pass |
| 3 | Not exist Email, Valid Password | “[rodriguez18@gmail.com](mailto:rodriguez18@gmail.com)”,  ”rodriguez” | Sign Up | Sign Up | Pass |
| 4 | Not exist Email, Invalid Password | “[anders@gmail.com](mailto:anders@gmail.com)”,  “thomas” | Not Sign Up | Sign Up | Fail |

**class** SignUpView(View):#[I]

form\_class = MyUserForm#[1]

template\_name = **'mycash/sign-up.html'**

*# display blank form*

**def** get(self, request):#[2]

form = self.form\_class(None)#[3]

**return** render(request, self.template\_name, {**'form'**: form})#[4]

*# process form data*

**def** post(self, request):#[5]

form = self.form\_class(request.POST)#[6]

**if** form.is\_valid():#[7]

user = form.save(commit=False)#[8]

*# cleaned (normalized) data*

email = form.cleaned\_data[**'email'**]

password = form.cleaned\_data[**'password'**]

user.set\_password(password)

user.save()#[9]

user = authenticate(email=email, password=password)#[10]

**if** user **is not** None:#[11]

login(request, user)#[12]

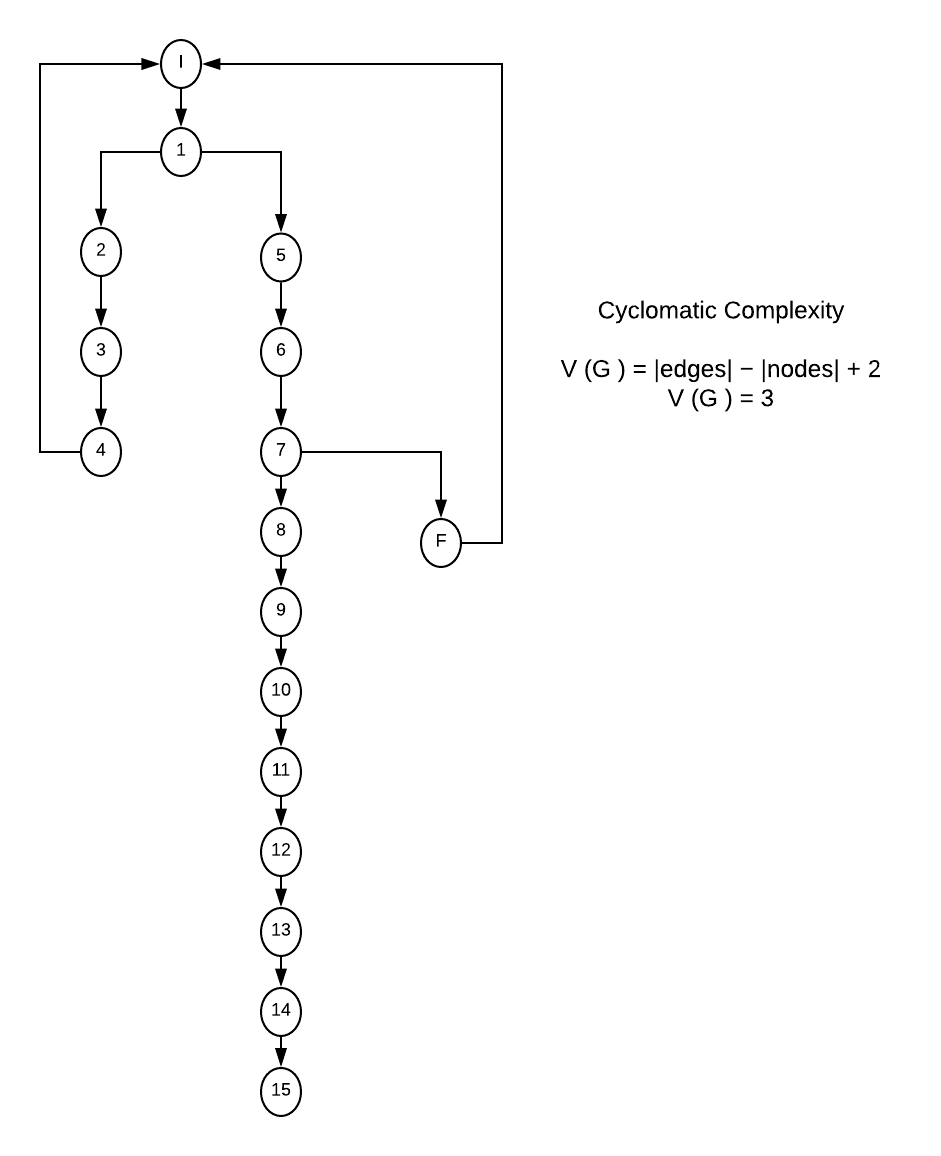
request.session[**'id'**] = user.id

db = DB()#[13]

db.create\_category(**'Other'**, user.id)#[14]

**return** redirect(**'mycash:overview'**)#[15]

**return** render(request, self.template\_name, {**'form'**: form})#[F]



### Sign In

|  |  |
| --- | --- |
| Test Case | |
| Test ID: TC-SI01 | Designed by: MyCash |
| Priority: High | D. Date: 03-06-18 |
| Test Title: Sign In | E. Date: 04-06-18 |
| Exec. By: Zevallos | |
| Classes Scope: LoginView - UserManager - User | |
| Environment: MyCash - Computer - DjangoServer - PyCharm | |
| Dependencies: User has a created account | |
| Related Cases: SignUpView – OverviewView - ModifyProfileView | |
| Objective: Allows Users to get into their account | |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| N | Test Steps | Test Data | Expected Result | Actual Result | Status |
| 1 | Provide Valid  Email, Valid Password | “p[maldonado@gmail.com](mailto:maldonado@gmail.com)”,  ”maldonado” | Sign In | Sign In | Pass |
| 2 | Provide Valid  Email, Invalid Password | “[juan@gmail.com](mailto:juan@gmail.com)”,  “juan” | Not Sign In | Sign In | Fail |
| 3 | Provide Invalid  Email, Valid Password | “[jacob@200](mailto:jacob@200)”17@gmail.com”,”jacob1996” | Not Sign In | Sign In | Fail |
| 4 | Provide Invalid  Email, Invalid Password | “#$!=@gmail.com”,  “#” | Not Sign In | Sign In | Fail |

**class SignInView(View):#[I]**

form = SignInForm()#[1]

**def** get(self, request):#[2]

context = {**'form'**: self.form} #[3]

**return** render(request, **'mycash/sign-in.html'**, context) #[4]

**def** post(self, request): #[5]

form = SignInForm(request.POST)#[6]

**if** form.is\_valid():#[7]

email = form.cleaned\_data[**'email'**]

password = form.cleaned\_data[**'password'**]

user = authenticate(email=email, password=password)#[8]

**if** user **is not** None:#[9]

**if** user.is\_active:#[10]

login(request, user)#[11]

request.session[**'id'**] = user.id

**return** redirect(**'mycash:overview'**)#[12]

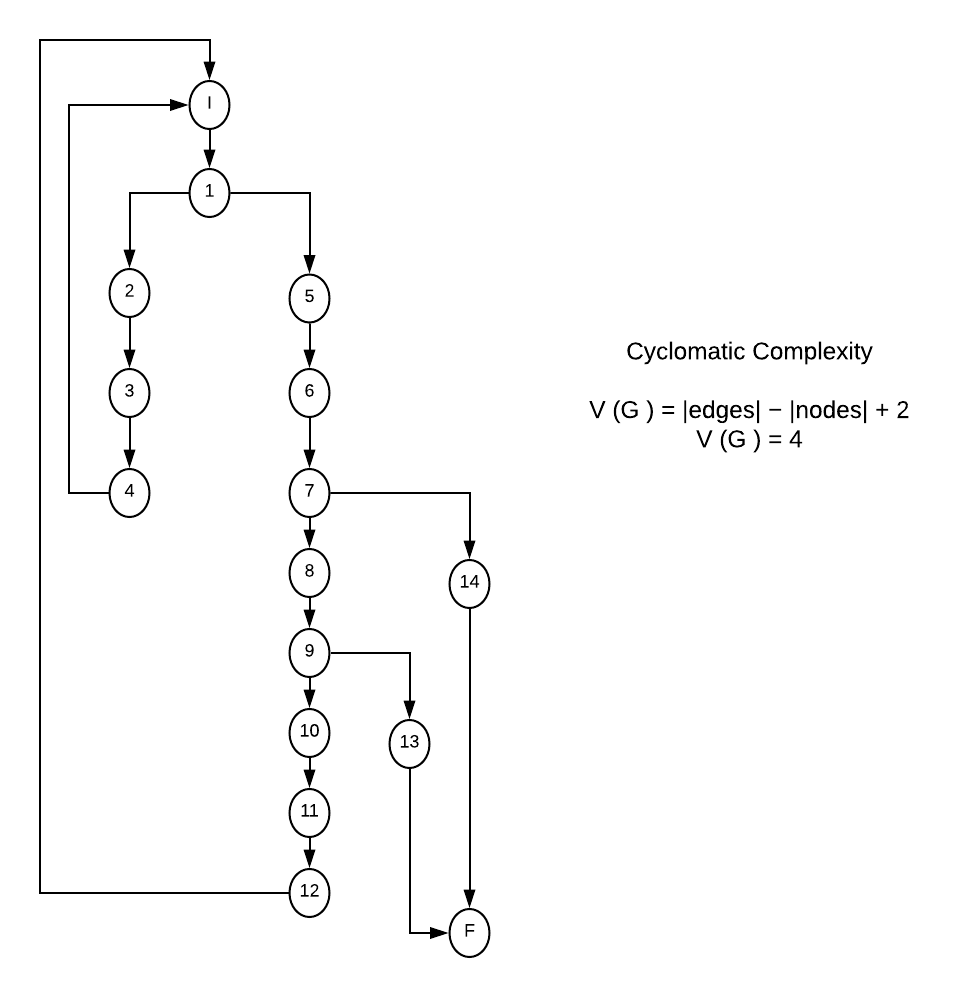
**else**:

context = {**'form'**: self.form, **'msg'**: **'User Is Not Active!'**}#[13]

**else**:

context = {**'form'**: self.form, **'msg'**: **'Error: Email - Password Invalid'**}#[14]

**return** render(request, **'mycash/sign-in.html'**, context)#[F]



### Add Category:

|  |  |
| --- | --- |
| Test Case | |
| Test ID: TC-MC\_AC01 | Designed by: MyCash |
| Priority: High | D. Date: 03-06-18 |
| Test Title: Add Category | E. Date: 04-06-18 |
| Exec. By: Zevallos | |
| Classes Scope: ManageCategoryView - CategoryManager – CategoryIncome - CategoryExpense | |
| Environment: MyCash - Computer - DjangoServer - PyCharm | |
| Dependencies: User must be logged in the application | |
| Related Cases: IncomeView – ExpenseView - HistoricalDataView | |
| Objective: Allows users to add their categories | |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| N | Test Steps | Test Data | Expected Result | Actual Result | Status |
| 1 | Provide Valid Name | “Home” | Add Category | Add Category | Pass |
| 2 | Provide Invalid Name | “!”#$%&” | Not Add Category | Add Category | Fail |

**class CategoryCreate(View):#[I]**

form\_class = CategoryForm#[1]

template\_name = **'mycash/manage-category.html'**

*# display blank form*

**def** get(self, request):#[2]

form = self.form\_class(None)#[3]

**return** render(request, self.template\_name, {**'form'**: form})#[4]

*# process form data*

**def** post(self, request):#[5]

form = self.form\_class(request.POST)#[6]

**if** form.is\_valid():#[7]

category = form.save(commit=False)#[8]

category.user = request.user

db = DB()#[9]

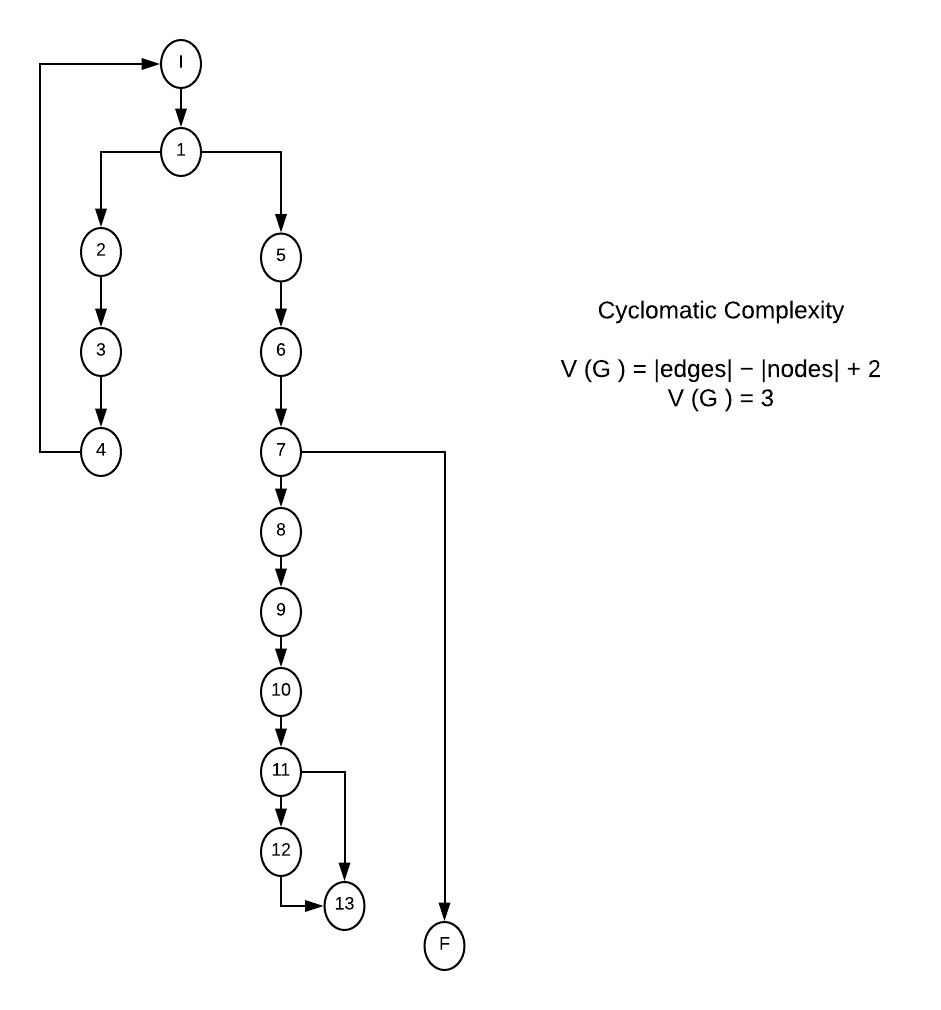
same = db.verify\_category(category.name, request.user.id)#[10]

**if not** same:#[11]

category.save()#[12]

**return** redirect(**'mycash:overview'**)#[13]

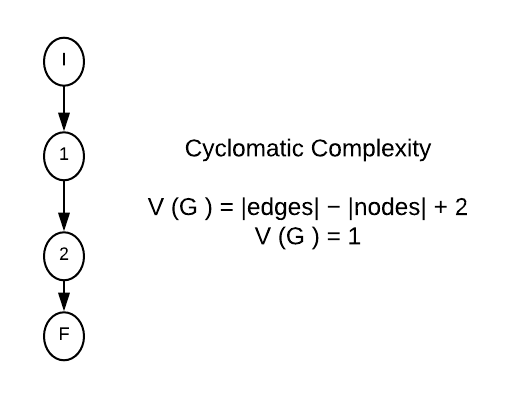
**return** render(request, self.template\_name, {**'form'**: form})#[F]



### Modify Category:

|  |  |
| --- | --- |
| Test Case | |
| Test ID: TC-MC\_MC01 | Designed by: MyCash |
| Priority: Low | D. Date: 03-06-18 |
| Test Title: Modify Category | E. Date: 04-06-18 |
| Exec. By: Zevallos | |
| Classes Scope: ManageCategoryView - CategoryManager – CategoryIncome - CategoryExpense | |
| Environment: MyCash - Computer - DjangoServer - PyCharm | |
| Dependencies: User must have at least one category | |
| Related Cases: IncomeView – ExpenseView - HistoricalDataView | |
| Objective: Allows users to modify their categories | |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| N | Test Steps | Test Data | Expected Result | Actual Result | Status |
| 1 | Provide Valid Name | “Health” | Modify Category | Modify Category | Pass |
| 2 | Provide Invalid Name | “>><<” | Not Modify Category | Modify Category | Fail |



**class CategoryUpdate(UpdateView):#[I]**

model = Category#[1]

template\_name = **'mycash/manage-category.html'**

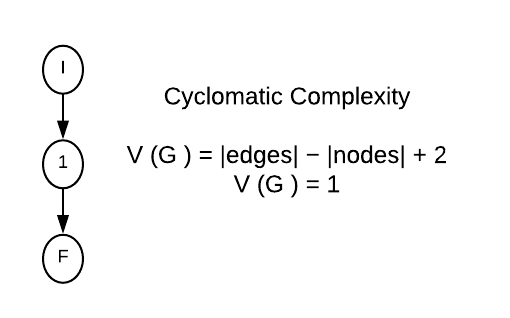
form\_class = CategoryForm#[2]

success\_url = reverse\_lazy(**'mycash:overview'**)#[F]

### Delete Category:

|  |  |
| --- | --- |
| Test Case | |
| Test ID: TC-MC\_DC01 | Designed by: MyCash |
| Priority: Low | D. Date: 03-06-18 |
| Test Title: Delete Category | E. Date: 04-06-18 |
| Exec. By: Zevallos | |
| Classes Scope: ManageCategoryView - CategoryManager – CategoryIncome - CategoryExpense | |
| Environment: MyCash - Computer - DjangoServer - PyCharm | |
| Dependencies: User must have at least one category | |
| Related Cases: IncomeView – ExpenseView - HistoricalDataView | |
| Objective: Allows users to delete their categories | |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| N | Test Steps | Test Data | Expected Result | Actual Result | Status |
| 1 | Click Delete | “Health” | Delete Category | Delete Category | Pass |

****

**class** CategoryDelete(DeleteView):#[I]

model = Category#[1]

success\_url = reverse\_lazy(**'mycash:overview'**)#[F]

### Add Income:

|  |  |
| --- | --- |
| Test Case | |
| Test ID: TC-MI\_AI01 | Designed by: MyCash |
| Priority: High | D. Date: 03-06-18 |
| Test Title: Add Income | E. Date: 04-06-18 |
| Exec. By: Zevallos | |
| Classes Scope: IncomeView - IncomeExpensesManager – Income | |
| Environment: MyCash - Computer - DjangoServer - PyCharm | |
| Dependencies: User must be logged in the application | |
| Related Cases: CategoryIncome – CategoryManager - ManageCategoryView - HistoricalDataView | |
| Objective: Allows users to add their income | |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| N | Test Steps | Test Data | Expected Result | Actual Result | Status |
| 1 | Provide Valid  Name, Valid Amount | “Salary”, “1000” | Add Income | Add Income | Pass |
| 2 | Provide Valid  Name, Invalid Amount | “Payments”,”-12” | Not Add Income | Add Income | Fail |
| 3 | Provide Invalid  Name, Valid Amount | “!”#$%&”,”1200” | Not Add Income | Add Income | Fail |
| 4 | Provide Invalid  Name, Invalid Amount | “<<->>”,”000” | Not Add Income | Add Income | Fail |

**class** IncomeCreate(View):#[I]

**def** get\_initial(self):#[1]

**return** {**'user'**: self.request.session[**'id'**]}#[2]

form\_class = IncomeForm#[3]

template\_name = **'mycash/manage-income.html'**

*# display blank form*

**def** get(self, request):#[4]

form = self.form\_class(request.user, None)#[5]

**return** render(request, self.template\_name, {**'form'**: form})#[6]

*# process form data*

**def** post(self, request):#[7]

form = self.form\_class(request.user, request.POST)#[8]

**if** form.is\_valid():#[9]

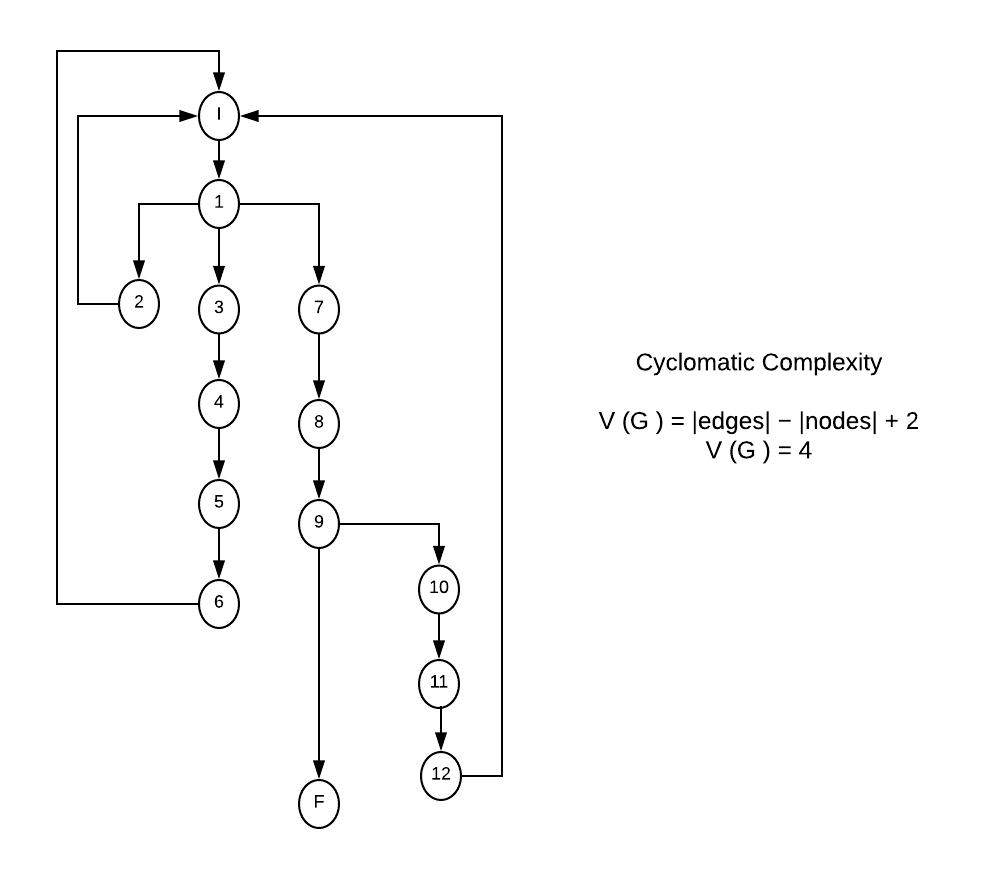
income = form.save(commit=False)#[10]

income.user = request.user

income.save()#[11]

**return** redirect(**'mycash:overview'**)#[12]

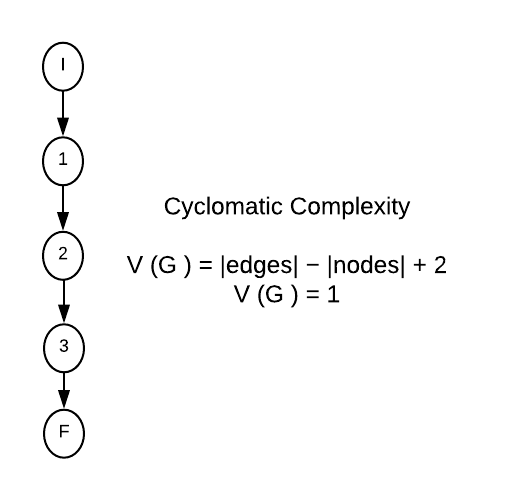
**return** render(request, self.template\_name, {**'form'**: form})#[F]



### Modify Income:

|  |  |
| --- | --- |
| Test Case | |
| Test ID: TC-MI\_MI01 | Designed by: MyCash |
| Priority: Low | D. Date: 03-06-18 |
| Test Title: Modify Income | E. Date: 04-06-18 |
| Exec. By: Zevallos | |
| Classes Scope: IncomeView - IncomeExpensesManager – Income | |
| Environment: MyCash - Computer - DjangoServer - PyCharm | |
| Dependencies: User must have at least one income | |
| Related Cases: CategoryIncome – CategoryManager - ManageCategoryView - HistoricalDataView | |
| Objective: Allows users to modify their income | |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| N | Test Steps | Test Data | Expected Result | Actual Result | Status |
| 1 | Provide Valid  Name, Valid Amount | “Job”, “2000” | Modify Income | Modify Income | Pass |
| 2 | Provide Valid  Name, Invalid Amount | “Bank”,”-0” | Not Modify Income | Modify Income | Fail |
| 3 | Provide Invalid  Name, Valid Amount | “-----”,”160” | Not Modify Income | Modify Income | Fail |
| 4 | Provide Invalid  Name, Invalid Amount | “;;;;;”,”-1000” | Not Modify Income | Modify Income | Fail |



**class** IncomeUpdate(UpdateView):#[I]

model = Income#[1]

form\_class = IncomeUpdateForm#[2]

template\_name = **'mycash/manage-income.html'**

**def** get\_success\_url(self):#[3]

**return** reverse(**'mycash:overview'**)#[F]

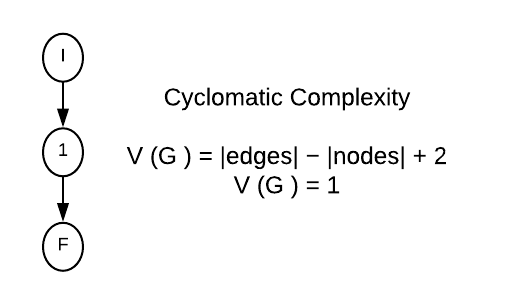
### Delete Income:

|  |  |
| --- | --- |
| Test Case | |
| Test ID: TC-MI\_DI01 | Designed by: MyCash |
| Priority: Low | D. Date: 03-06-18 |
| Test Title: Delete Income | E. Date: 04-06-18 |
| Exec. By: Zevallos | |
| Classes Scope: IncomeView - IncomeExpensesManager – Income | |
| Environment: MyCash - Computer - DjangoServer - PyCharm | |
| Dependencies: User must have at least one income | |
| Related Cases: CategoryIncome – CategoryManager - ManageCategoryView - HistoricalDataView | |
| Objective: Allows users to delete their income | |

**class** IncomeDelete(DeleteView):#[I]

model = Income#[1]

success\_url = reverse\_lazy(**'mycash:overview'**)#[F]



### Add Expense:

|  |  |
| --- | --- |
| Test Case | |
| Test ID: TC-ME\_AE01 | Designed by: MyCash |
| Priority: High | D. Date: 03-06-18 |
| Test Title: Add Expenses | E. Date: 04-06-18 |
| Exec. By: Zevallos | |
| Classes Scope: ExpenseView - IncomeExpensesManager – Expense | |
| Environment: MyCash - Computer - DjangoServer - PyCharm | |
| Dependencies: User must be logged in the application | |
| Related Cases: CategoryExpense – CategoryManager - ManageCategoryView - HistoricalDataView | |
| Objective: Allows users to add their expenses | |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| N | Test Steps | Test Data | Expected Result | Actual Result | Status |
| 1 | Provide Valid  Name, Valid Amount | “Fruits”, “120” | Add Expense | Add Expense | Pass |
| 2 | Provide Valid  Name, Invalid Amount | “Vegetables”,”-200” | Not Add Expense | Add Expense | Fail |
| 3 | Provide Invalid  Name, Valid Amount | “°°°°”,”300” | Not Add Expense | Add Expense | Fail |
| 4 | Provide Invalid  Name, Invalid Amount | “////”,”000” | Not Add Expense | Add Expense | Fail |

### Modify Expense:

|  |  |
| --- | --- |
| Test Case | |
| Test ID: TC-ME\_ME01 | Designed by: MyCash |
| Priority: Low | D. Date: 03-06-18 |
| Test Title: Modify Expense | E. Date: 04-06-18 |
| Exec. By: Zevallos | |
| Classes Scope: ExpenseView - IncomeExpensesManager – Expense | |
| Environment: MyCash - Computer - DjangoServer - PyCharm | |
| Dependencies: User must have at least one expense | |
| Related Cases: CategoryExpense – CategoryManager - ManageCategoryView - HistoricalDataView | |
| Objective: Allows users to modify their expenses | |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| N | Test Steps | Test Data | Expected Result | Actual Result | Status |
| 1 | Provide Valid  Name, Valid Amount | “Dessert”, “2000” | Modify Expense | Modify Expense | Pass |
| 2 | Provide Valid  Name, Invalid Amount | “Bread”,”-45” | Not Modify Expense | Modify Expense | Fail |
| 3 | Provide Invalid  Name, Valid Amount | “¿¿¿”,”160” | Not Modify Expense | Modify Expense | Fail |
| 4 | Provide Invalid  Name, Invalid Amount | “{}”,”0” | Not Modify Expense | Modify Expense | Fail |

### Delete Expense:

|  |  |
| --- | --- |
| Test Case | |
| Test ID: TC-ME\_DE01 | Designed by: MyCash |
| Priority: Low | D. Date: 03-06-18 |
| Test Title: Delete Expense | E. Date: 04-06-18 |
| Exec. By: Zevallos | |
| Classes Scope: ExpenseView - IncomeExpensesManager – Expense | |
| Environment: MyCash - Computer - DjangoServer - PyCharm | |
| Dependencies: User must have at least one expense | |
| Related Cases: CategoryExpense – CategoryManager - ManageCategoryView - HistoricalDataView | |
| Objective: Allows users to delete their expense | |

### Log Out

|  |  |
| --- | --- |
| Test Case | |
| Test ID: TC-LU01 | Designed by: MyCash |
| Priority: High | D. Date: 03-06-18 |
| Test Title: Log Out | E. Date: 04-06-18 |
| Exec. By: Zevallos | |
| Classes Scope: OverviewView - UserManager - User | |
| Environment: MyCash - Computer - DjangoServer - PyCharm | |
| Dependencies: User must be logged in the application | |
| Related Cases: IndexView | |
| Objective: Takes the user out of the session and the next time he wants to log in, he will ask for his password | |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| N | Test Steps | Test Data | Expected Result | Actual Result | Status |
| 1 | Click Log out | “[jesamin@gmail.com](mailto:jesamin@gmail.com)” | Log Out | Log Out | Pass |

### Delete User Account:

|  |  |
| --- | --- |
| Test Case | |
| Test ID: TC-DUA01 | Designed by: MyCash |
| Priority: Low | D. Date: 03-06-18 |
| Test Title: Delete user account | E. Date: 04-06-18 |
| Exec. By: Zevallos | |
| Classes Scope: User | |
| Environment: MyCash - Computer - DjangoServer - PyCharm | |
| Dependencies: The administrator must logged in the system | |
| Objective: Modify the status of a user | |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| N | Test Steps | Test Data | Expected Result | Actual Result | Status |
| 1 | Click Active | “jesamin@gmail.com” | Delete Account | Delete Account | Pass |

### Delete Account:

|  |  |
| --- | --- |
| Test Case | |
| Test ID: TC-DA01 | Designed by: MyCash |
| Priority: Low | D. Date: 03-06-18 |
| Test Title: Delete account | E. Date: 04-06-18 |
| Exec. By: Zevallos | |
| Classes Scope: ModifyProfileView – UserManager - User | |
| Environment: MyCash - Computer - DjangoServer - PyCharm | |
| Dependencies: The administrator must logged in the system | |
| Related Cases: IndexView | |
| Objective: Inactivates your account | |

### See Technical Request:

|  |  |
| --- | --- |
| Test Case | |
| Test ID: TC-SeTR01 | Designed by: MyCash |
| Priority: High | D. Date: 03-06-18 |
| Test Title: See Technical Request | E. Date: 04-06-18 |
| Exec. By: Zevallos | |
| Classes Scope: UserManager – Administrator | |
| Environment: MyCash - Computer - DjangoServer - PyCharm | |
| Dependencies: The administrator must logged in the system | |
| Related Cases: | |
| Objective: The request pool is a place where all technical request are stored. There is just one request pool and only administrators will have the access to it | |

### Send Technical Request:

|  |  |
| --- | --- |
| Test Case | |
| Test ID: TC-SdTR01 | Designed by: MyCash |
| Priority: Low | D. Date: 03-06-18 |
| Test Title: Send Technical Request | E. Date: 04-06-18 |
| Exec. By: Zevallos | |
| Classes Scope: OverviewView - UserManager – User | |
| Environment: MyCash - Computer - DjangoServer - PyCharm | |
| Dependencies: The user must be logged in the system. | |
| Related Cases: | |
| Objective: Sends a request to the administrator in case of any problem. | |

# EXTRA DOCUMENTATION

## Meetings:

| Meeting Agenda [M3\_01] | |
| --- | --- |
| Date: | 15/05/2018 |
| Time:  Type: | 10:00 – 11:30 am  Exhaustive Meeting |

**Agenda details:**

**Define new roles.**

Selects a different person in a role for the second part of the SPFG project. Selection was defined as follow:

* Project Manager:
  + Alonso Jesús Cerpa Salas -> José David Mamani Vilca.
* Developer:
  + Percy Maldonado Quispe -> Alonso Jesús Cerpa Salas
* Designer and Analyst:
  + José David Mamani Vilca -> Percy Maldonado Quispe.

**Use Case Diagram observation.**

Some use cases were under observation since the time to deadline is pretty near. It was decided to erase two of them and left the others under observation:

* [Erased] Send Message (UC-SM).
* [Under Observation] Search User (UC-SU).
* [Under Observation] Personalize (UC-P).

Erased use cases will affect subsequent diagrams (Such as Sequence Diagrams, Class Diagram, Entity Diagram, etc.). So an upgrade in those diagrams was designated to the designer.

Different function names in code, class diagram and sequence diagram.

A standardization was talked between designer and developer. Function names will be change in code and class diagram.

| Meeting Agenda [M3\_02E] | |
| --- | --- |
| Location: | San Pablo Catholic University |
| Date: | 17/05/2018 |
| Time:  Type: | 10:00 – 10:30 am  Emergency Meeting |

**Agenda details:**

**New member in MyCash development team. Reorganizing roles and work plan.**

* Presenting a new member: Jesamin Zevallos Quispe.
* Assigning a role.
  + [Conclusion] Analyst: Jesamin Zevallos Quispe.
* Applying a help routine with the new member.
* Sharing repository. Creating a new branch for the new member.
* Work Plan and Roles distribution need to be restructured.
  + [Conclusion]:
  + Project Manager: José David Mamani Vilca.
  + Developer: Alonso Jesús Cerpa Salas.
  + Designer: Percy Maldonado Quispe.
  + Analyst: Jesamin Zevallos Quispe.

| Meeting Agenda [M3\_03] | |
| --- | --- |
| Location: | San Pablo Catholic University |
| Date: | 24/05/2018 |
| Time:  Type: | 10:00 – 11:30 am  Exhaustive Meeting |

**Agenda details:**

**Preparing Basic and Complex Tests for MyCash.**

Different use cases requires different tests to be proved.

Choosing between “Black Box” and “White Box” according to the use case.

Selecting a specific software to make massive test in different use cases.

Test Graph.

**Discarding some use cases.**

Use Case “Payment Plan” was discarded from MyCash system.

Updating Risk Management.

| Meeting Agenda [M3\_03] | |
| --- | --- |
| Location: | San Pablo Catholic University |
| Date: | 06/06/2018 |
| Time:  Type: | 10:00 – 1:30 am  Meeting |

**Agenda details:**

**Reviewing missing documentation**

* Different diagrams were outdated.
* Updating Case Use diagram, Sequence diagram, Entity Relationship diagram, Relational Diagram and Class Diagram.
* Correcting some grammar mistakes in Case Use Details.
* Reviewing Component Diagram and Deployment Diagram.
* Test documentation.

**Next Season.**

* Updating work plan for next season.
* Re-organizing work plan due to a new member in third season.

1: Se pierde la barra en la pantalla de inicio. \*

2. Login: Evitar que el scroll gire. \*

3. Iniciar sesión con categorías definidas. \*

4. Validación de Correo. \*

5. Buscador en incomes y expenses. Nombre o Fecha. \*

6. Añadir ingresos en la misma barra que añadir metas. \*

7. Añadir ingreso con el botón añadir categoría.

8. Delete account con confirmación de passsword.

9. Limpiar los menús emergentes.(Cookies)

10. Lista de categorías: Mostrar Total de la categoría.

11. Balance de metas con los ingresos y los egresos.