**All Units:**

1. Scenario 1 (for understanding & testing)
2. Database TABLES
3. User ACTIONS/TRANSACTIONS

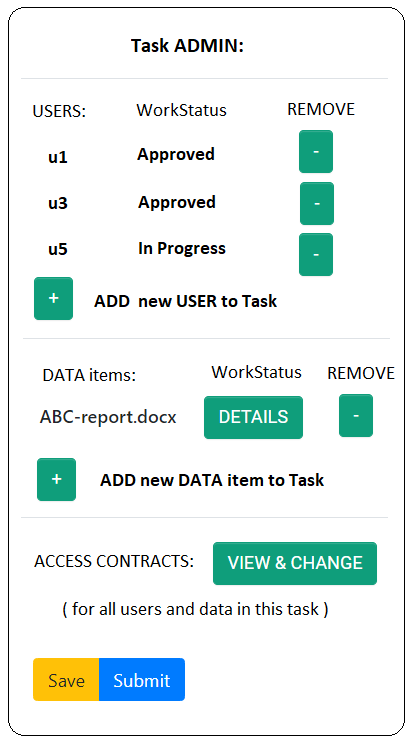
# please **integrate** the following into your document, under each section please **keep the format I provide here** as we go, you should also make changes in the rest of your document so that they reflect the guidance here. Also have your tables and diagrams here in, as much as possible, **editable** format so that we can easily make changes later #

**(1). Scenario 1 (for understanding & testing)**

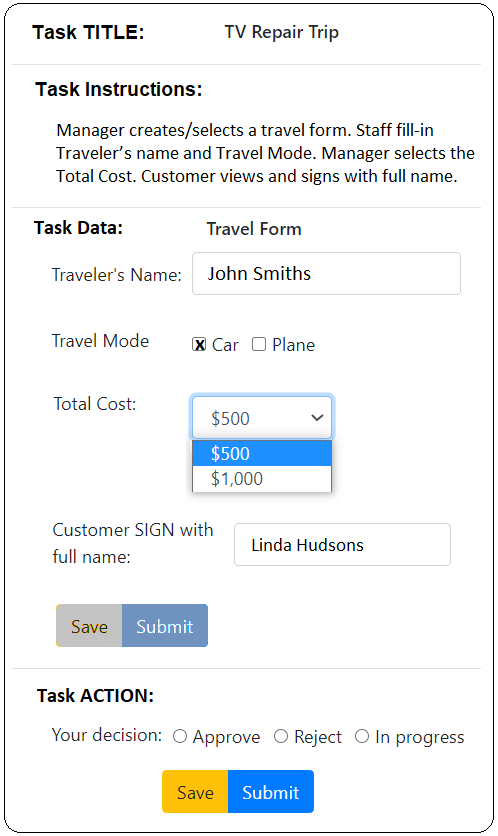
Here is a scenario to help you to understand and test our system. The system should work with many/any scenarios.

USERS: we have 6 users: u1,u2,u3,u4,u5,u6.

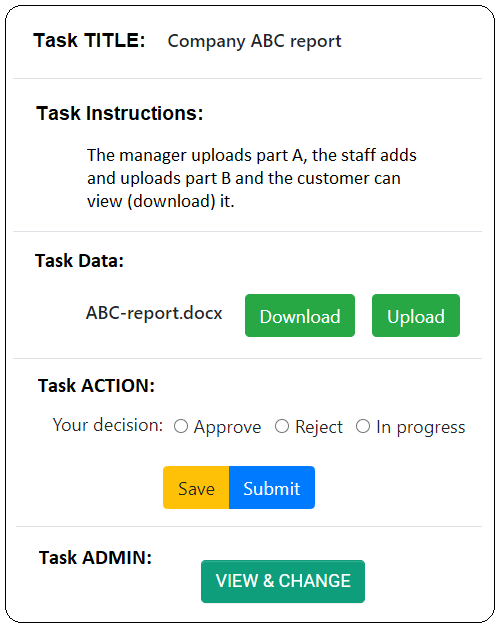
In the very beginning unit A should let these users register, login, assign and be assigned roles. Currently these users are assigned the following roles: u1 and u2 are managers, u3 and u4 are staffs, u5 and u6 are customers.



Picture 3



Picture 2



Picture 1

JOB/TASKS: then, some user create a job **Jn** (by unit B) with two tasks:

* task **Tx** is created where u1,u3,u5 are added as participants using policy P1, and a **file Fs** is added
* task **Tc** is created with u2,u4,u6 as participants using policy P2, and a **form Fv** is added

In **task Tx**:

The manager starts with uploading part A of **file Fs** for the company ABC report, then the staff adds(upload) part B, and the customer can view (download) it. Then each of them can approve or reject it. The task is done once each participant/user submit the decision. Each should see a web page similar to what is in Picture 1.

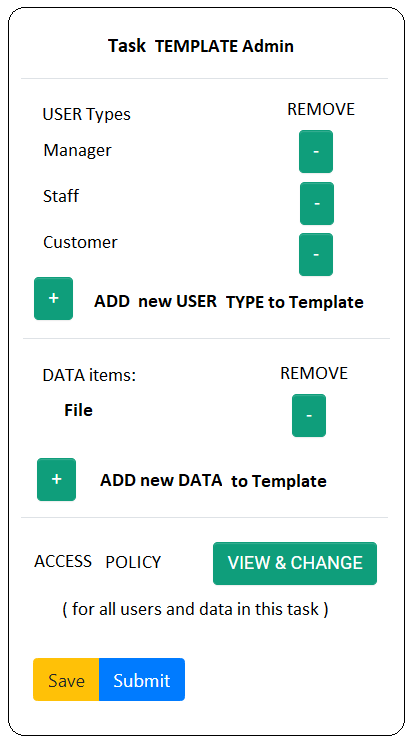
In **task Tc**:

The manager creates/selects a travel **form**. Staff fill-in Element A: Traveler’s name and Element B: Travel Mode. Manager selects Element C: the Total Cost. Customer views and signs Element D. Then, each of them can approve or reject it. The task is done once each participant/user submit the decision. Each should see a web page similar to what is in Picture 2.

SAVE button will save the selection to DB table. SUBMIT button will save and lock the current task for the given user (can’t go back and change things again, can still view it though). Unit B is responsible for the whole task/job page but Unit C and Unit D takes care of the Task Data section. If the users have appropriate access level (u1 and u2 in this case) then they’ll also see the “Task Admin” section (done by UnitB) where they can add/delete users and data items as shown in Picture 3. Under Data items DETAILS button will show the work status(approved or done?) of the data item by each user/participating based on taskPart\_status from taskDetails\_T. In this example manager users can also see more details of the ACCESS contracts, activate/deactivate and lock/unlock the data items by clicking on the “VIEW & CHANGE” button (done by unit A/C).

CREATE a NEW taskTEMPLATE

1) FORM



Picture 4

|  |  |  |  |
| --- | --- | --- | --- |
| policy\_id | task\_type | task\_title | task\_instructions |
| 02 | 2 | ABC company report | The manager uploads part A, the staff adds and upload part B and customer can view (download) it. |

Add a new row to taskTemplate\_T

Nothing is added to **taskTemplateDetails\_T, yet**

2) FORM

Ask if this user (who’s making a taskTemplate) would like to:

+ ADD a USER ?

exp: add a user type manager: 2 => adding a new ROW in **taskTemplateDetails\_T**

<check if this task has any DATA item ?>

|  |  |  |
| --- | --- | --- |
| taskTemplate\_id | userRole\_id | dataType\_id |
| 001 | 2 |  |

exp: add a user type manager: 3 => adding a new ROW in **taskTemplateDetails\_T**

|  |  |  |
| --- | --- | --- |
| taskTemplate\_id | userRole\_id | dataType\_id |
| 001 | 2 |  |
| 001 | 3 |  |

<check if this task has any DATA item ?>

+ ADD a DATA ?

exp: add a DATA item type: **4** => insert it to the DATA fields of all ROWS of THIS taskTemplate (id=001) in **taskTemplateDetails\_T**

|  |  |  |
| --- | --- | --- |
| taskTemplate\_id | userRole\_id | dataType\_id |
| 001 | 2 | **4** |
| 001 | 3 | **4** |

+ ADD a USER ?

exp: add a user type manager: **4** => adding a new ROW in **taskTemplateDetails\_T**

<check if this task has any DATA item ? YES>

|  |  |  |
| --- | --- | --- |
| taskTemplate\_id | userRole\_id | dataType\_id |
| 001 | 2 | 4 |
| 001 | 3 | 4 |
| 001 | 4 | 4 |

+ ADD a DATA ?

exp: add a DATA item type: **5** => insert it to the DATA fields of all ROWS of THIS taskTemplate (id=001) in **taskTemplateDetails\_T**

<check if this task has any DATA item ? YES => this new DATA in NOT the first >

<check how many users in this taskTemplate? => 3 user types **2,3,4** > => add a **new ROW for EACH of these users** for this NEW DATA item;

|  |  |  |
| --- | --- | --- |
| taskTemplate\_id | userRole\_id | dataType\_id |
| 001 | 2 | 4 |
| 001 | 3 | 4 |
| 001 | 4 | 4 |
| 001 | 2 | 5 |
| 001 | 3 | 5 |
| 001 | 4 | 5 |

POLICIES: in the system, the policies P1 and P2 are stored in the policy\_T and policyDetails\_T tables:

=============================

Policy P1 = in each task,

+ manager can view, modify, add a new data item, and delete it;

+ staff can view(download) and modify(upload)

+ customer can view (download)

+ only manager can send messages to other and delete messages, all can read/view messages

-------------------------------------------------------

Policy P2 = in each task,

+manager can view, modify, add a new data item, and delete it, activate and deactivate it;

+staff can view, modify, add a new data item, and delete it;

+customer can view, modify

+everyone can send/read messages to each other but only manager can delete messages

================================

When a request to create this job **Jn** (which contains tasks Tx and Tc) is made (by some user) unit B needs to create that new job Jn (based on a template), unit B should let the user (the manager u1 and u2 for example) to choose whether:

1. to create a new job template or
2. get a prebuilt job template from the list/library of existing templates in the system

If (a) then unit B needs to create a job template by adding task templates to it. Therefore, it should let users create task templates. Each task template can be built by adding:

* user types to it
* data types to it
* create a new policy or select an existing policy [this is done by unit A or D]

Once a task template is finalized/saved it will be stored in taskTemplate\_T . Now it can be used to create a real task.

Once a job template is finalized/saved it will be stored in jobTemplate\_T . Now it can be used to create a real job.

If (b) then a real job can be generated based on a pre-built job template by copying fields from jobTemplate\_T into job\_T and replace a user role = specific userID, a data type = specific data\_id similarly with all the tasks in it. Unit B should let the users (if permitted ) be able to view/delete/edit: task templates, job templates, tasks, and jobs.

DATA ITEMS:

When unit B lets users to create a job, at some point a data item is needed and should to be created. First it will go to unit C.

+ unit C, for this matter, needs to:

(i) see first what role each participating user has (from user\_T) at that moment, then

(ii) use what is in the policy tables for policy P1/P2 to

(iii) create a contract which is to apply rules in policy P1/P2 to these SPECIFIC users and data items = copy what is in {policy\_T and policyDetails\_T} into {contract\_T and contractDetails\_T}.

As the results, these two contracts are created and added to contract\_T and contractDetails\_T tables:

----------------------------------

Contract Cw1: (replace manager=u1, staff=u3, customer=u5, data item=**file** Fs where data\_id=104)

+u1 can create, read/view, modify(replace=upload), delete file Fs;

+u3 can read, modify Fs

+u5 can read Fs

+u1 has access to MessageSender (can send messages to others) and can delete messages, u3 and u5 can read messages

-------------------------------------

Contract Cw2: (replace manager=u2, staff=u4, customer=u6, data item= **form** Fv where data\_id=105)

+u2 can read, modify, add a new data item, and delete Fv, activate and deactivate it;

+u4 can read, modify, add a new data item, and delete Fv;

+u6 can read, modify Fv

+u2,u4,u6 have access to MessageSender (can send messages to others) but only u2 can delete messages,

------------------------------------

Unit C should be able to let:

+ only manager u1 be able to add a data item to task Tx, select it as a file, then have options to

* upload the file OR
* insert a pre uploaded file (from files u1 has access to)

+ only manager u2 be able to add a data item to task Tc, select it as a form, then the following will be taken cared by unit D which provide these options to

* create a new form
* insert a prebuilt form (from forms u2 has access to)

Unit C should also be able to let these managers change the contracts for each of the users participating in the tasks they are the manager for. Other users can view their contracts but not being able to change them.

FORMS:

If the given data item is a form then that’s when unit D takes over. In unit D: the format and contents of forms are stored using **a combination of MySQL tables and files**.

Unit D should give users these options:

1. to create a new form or
2. get a copy of a prebuilt form from the list/library of existing forms in the system

Whether it’s a form template or a live form, the users can start with an empty form and keep adding elements/parts/components to it until it’s done.

The set of possible web elements/parts/components of a form (SwE) is: { label, textboxes, radio buttons, check boxes, dropdowns, and buttons }. There are standard and GENERIC codes for each type of these form parts/components which are stored in files.

Then, unit D needs to be able to dynamically compose and print out the HTML/PHP codes for the whole (any) form and save it by combining the codes of all of its parts/components.

For example, in **task Tc**: the manager generates a travel form Fv (id=001) based on a form template (id=0001 in **formTemplate\_T**) which has:

* Element #1: label for Form title;
* Element #2: textbox for Traveler’s name;
* Element #3: radio buttons (of 2) for Travel Mode;
* Element #4: select/dropdowns (of 2) for the Total Cost.
* Element #5: textbox for for the Customer to sign

Let’s see how this form template was created, by some user. Each time when an element, for example Element #2, is added to this template by that user, unit D will need to:

1. Has a form for this user to select one specific type of elements in SwE (in this example it’s a textbox) and also enter necessary information, such as size of element, the Label value, other variable NAMES and VALUES in the element, so that it can use those to REPLACE the generic variable NAMES and VALUES in the standard code (to help create php codes for the form and its PHP server program for the whole form later) EXAMPLE: https://www.phpformbuilder.pro/drag-n-drop-form-builder/index.html
2. reads the standard codes for that element (textbox) from “/form/elements/formTemplateElement\_2.txt” in **formElementType\_T** , then copy those codes with the REPLACEMENTS, using the info above in (a), into the file that holds the codes for that element in the form template:” /form/ temp/formT\_001\_p2.txt” in **formTemplateDetails\_T**.

Shortly speaking, for each element of a form:

[GENERIC standard codes] + [ REPLACEMENT of generic variable NAMES and VALUES with SPECIFIC ones] = [html/php codes for the given element]

For example, for element #2:

|  |  |
| --- | --- |
| /form/elements/formTemplateElement\_2.txt |  |
| <label for="**xxx**">**xxx\_label**:</label><br> <input type="text" id="xxx" name="xxx"><br> | 🡸 xxx is the GENERIC name of TEXTBOX variable  xxx\_label is a GENERIC value in the STANDARD codes |

Will lead to:

|  |  |
| --- | --- |
| /form/ temp/formT\_001\_p2.txt |  |
| <label for="t1">**Traveler’s Name:</**label><br> <input type="text" id="t1" name="t1"><br> | 🡸 xxx = **t1**  xxx\_label = **Traveler’s Name** (for THIS SPECIFIC form template) |

So, basically, unit D needs to:

1. build GENERIC CODES for all types of elements/parts/components in **formElementType\_T**  and store them in those files.
2. then, unit D needs to have its own codes which can ask users what kinds of elements/parts/components they want to have in the form and now generate specific codes for each of those elements and therefore get the codes for the whole form template and store it in a file in **formTemplate\_T** or **formTemplateDetails\_T**.
3. When users create a live form based on a form template unit D needs to copy the codes from the files in **formTemplate\_T** or **formTemplateDetails\_T** to files in **formDetails\_T** or **form\_T** . The PHP codes which process the form should also be defined. Unit D should also make sure that it can store the current values of fields/variables in the form in the files. The format of those files can be JSON or just TEXT.

MESSAGES:

Unit E needs to look at the contracts (in contract\_T and contractDetails\_T) to see who can send/view/delete messages; then let the users compose/send, view/read, or delete messages (if allowed) as listed in the list of actions. All messages are stored in **message\_T**. Compose/create a new message means adding a new row to **message\_T**.

**(2). Database TABLES**

**2.1 List of (Editable) Tables:**

(\*) indicates that other units are responsible for the creation of the given table but it may be beneficial for your unit to know what’s there or you may need to use/access it.

|  |  |  |
| --- | --- | --- |
| **userRole\_T** | (\*) |  |
| userRole\_id | userRole\_title | userRole\_info |
| 1 | Admin | Person with highest access rights |
| 2 | Manager | Leader of a group or division of the company |
| 3 | Staff | Member of a group or division of the company |
| 4 | Customer | Customer of the company |
| 5 | Public | Anyone who is not login-ed |
| 6 | Any | Any of the above |

|  |  |  |
| --- | --- | --- |
| **userStatus\_T** | (\*) |  |
| userStatus\_id | userStatus\_title | userStatus\_info |
| 1 | Active | for users who are registered, approved, and can function in the system |
| 2 | Passive | for users who are registered, but not approved yet, or put on paused |
| 3 | Terminated | for users who are registered but no longer in the system |

# Extra status can be added if we need it

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **user\_T** | (\*) |  |  |  |  |  |  |
| user\_id | user\_role | user\_loginName | user\_pass | user\_fullName | user\_email | user\_info | user\_status |
| 001 | 1 | u0 | p0 | n0 | e0 | i0 | 1 |
| 002 | 2 | u1 | p1 | n1 | e1 | i1 | 1 |
| 003 | 2 | u2 | p2 | n2 | e2 | i2 | 1 |
| 004 | 3 | u3 | p3 | n3 | e3 | i3 | 1 |
| 005 | 3 | u4 | p4 | n4 | e4 | i4 | 1 |
| 006 | 4 | u5 | p5 | n5 | e5 | i5 | 1 |
| 007 | 4 | u6 | p6 | n6 | e6 | i6 | 1 |
| 008 | 4 | u7 | p7 | n7 | e7 | i7 | 2 |

|  |  |  |
| --- | --- | --- |
| **dataType\_T** | (\*) |  |
| dataType\_id | dataType\_title | dataType\_info |
| 1 | Any | Any data item |
| 2 | Self | Data item created/owned by the same user |
| 3 | Other | Data item created by the other users |
| 4 | Form | A web form to be filled-in |
| 5 | File | A file attached to a task or owned by a user |
| 6 | Database | Fields, tables, in database |
| 7 | MessageSender | Where IN messages are accepted |
| 8 | Task | A task |
| 9 | Job | A job |

|  |  |  |
| --- | --- | --- |
| **dataStatus\_T** | (\*) |  |
| dataStatus\_id | dataStatus\_title | dataStatus\_info |
| 0\*\*\* | Terminated | for data which was created but no longer available (deleted) in the system |
| 1\*\*\* | Alive | for data which was created and available in the system |
| 11\*\* | Approved | for data which is approved/unlocked and is ready to be used in the system |
| 10\*\* | Rejected/Unapproved | for data which was created but not approved yet |
| 1\*1\* | Released/Unlocked | for data which was is available to be used in the system |
| 1\*0\* | Locked | not available for others to use besides the user who is currently using it or last modified it |
| 1\*\*1 | Active | for data which is visible (activated) in the system |
| 1\*\*0 | Passive | for data which is not visible (deactivated) in the system |

# status consists of 4 digits \*\*\*\*, you’ll need to split it into 4 single digits to find out about the status OR you can split this into 4 columns

# active means the given data item is visible; locked means it’s not available for others to use besides the user who is currently using it or last modified it

# approved means it’s been checked/verified and ready to be used; a data item can only be used if it’s alive, active, approved and unlocked, i.e. 1111

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **data\_T** | (\*) |  |  |  |  |  |  |
| data\_id | data\_status | data\_location | data\_type | data\_modifier | data\_changed | data\_owner | data\_created |
| 0001 | 1111 | user|user\_info | 6 | 001 | 2025-01-19 05:10:28 | 001 | 2019-01-19 05:10:28 |
| 0002 | 1111 | SendMessage.php | 7 | 001 | 2025-01-19 05:10:28 | 001 | 2019-01-19 05:10:28 |
| 0104 | 1111 | /upload/ABC-report.docx | 5 | 002 | 2035-01-19 09:13:07 | 002 | 2019-01-19 09:13:07 |
| 0105 | 1111 | /form/form1.json | 4 | 003 | 2025-01-19 05:10:28 | 003 | 2019-01-19 05:10:28 |

# data\_location for type 6: <table>|<field>; for type 4: name of file which contains the web form data in TEXT or JSON

|  |  |  |
| --- | --- | --- |
| **policyStatus\_T** | (\*) |  |
| policyStatus\_id | policyStatus\_title | policyStatus\_info |
| 1 | Active | policy which is approved and be applied in the system |
| 2 | Passive | policy which is not approved yet or put on paused |
| 3 | Terminated | policy which is no longer in the system |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **policy\_T** | (\*) |  |  |  |  |
| policy\_id | policy\_title | policy\_owner | policy\_status | policy\_created | policy\_expired |
| 01 | X | 001 | 2 | 2010-01-19 03:14:07 | 2050-01-19 03:14:07 |
| 02 | P1 | 002 | 1 | 2018-01-19 05:10:28 | 2025-01-19 05:10:28 |
| 03 | P2 | 003 | 1 | 2018-01-19 09:13:07 | 2035-01-19 09:13:07 |

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **policyDetails\_T** | (\*) |  |  |  |  |  |  |  | |  | |
| policy\_id | userRole\_id | dataType\_id | a1\_read | a2\_write | a3\_create | a4\_delete | a5\_approve | a6\_reject | a7\_activate | | a8\_deactivate | | a9\_lock | a10\_unlock |
| 01 | 6 | 2 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | | 1 | | 1 | 1 |
| **02** | **2** | **1** | **1** | **1** | **1** | **1** | **1** | **1** | **0** | | **0** | | **1** | **1** |
| **02** | **3** | **1** | **1** | **1** | **0** | **0** | **0** | **0** | **0** | | **0** | | **0** | **0** |
| **02** | **4** | **1** | **1** | **0** | **0** | **0** | **0** | **0** | **0** | | **0** | | **0** | **0** |
| 02 | 2 | 7 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | | 0 | | 0 | 0 |
| 02 | 3 | 7 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | | 0 | | 0 | 0 |
| 02 | 4 | 7 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | | 0 | | 0 | 0 |
| **03** | **2** | **1** | **1** | **1** | **1** | **1** | **1** | **1** | **1** | | **1** | | **0** | **0** |
| **03** | **3** | **1** | **1** | **1** | **0** | **0** | **1** | **1** | **0** | | **0** | | **0** | **0** |
| **03** | **4** | **1** | **1** | **1** | **0** | **0** | **0** | **0** | **0** | | **0** | | **0** | **0** |
| 03 | 2 | 7 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | | 0 | | 0 | 0 |
| 03 | 3 | 7 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | | 0 | | 0 | 0 |
| 03 | 4 | 7 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | | 0 | | 0 | 0 |

# policy P1 is implemented by three green rows, policy P2 is implemented by three blue rows

# in the columns for actions: 1 means “can”; 0 means “can NOT”

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **contract\_T** | (\*) |  |  |  |  |  |  |
| contract\_id | task\_id | contract\_creater | contract\_status | contract\_title | policy\_id | contract\_created | contract\_expired |
| 001 | 001 | 002 | 1 | Cw1 | 02 | 2019-01-19 05:10:28 | 2025-01-19 05:10:28 |
| 002 | 002 | 003 | 1 | Cw2 | 03 | 2019-01-19 09:13:07 | 2035-01-19 09:13:07 |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **contractDetails\_T** | (\*) |  |  |  |  |  |  | | |  | |  | |
| contract\_id | user\_id | data\_id | a1\_read | a2\_write | a3\_create | a4\_delete | a5\_approve | a6\_reject | a7\_activate | | a8\_deactivate | | a9\_lock | a10\_unlock |
| **001** | **002** | **104** | **1** | **1** | **1** | **1** | **1** | **1** | **0** | | **0** | | **1** | **1** |
| **001** | **004** | **104** | **1** | **1** | **0** | **0** | **0** | **0** | **0** | | **0** | | **0** | **0** |
| **001** | **006** | **104** | **1** | **0** | **0** | **0** | **0** | **0** | **0** | | **0** | | **0** | **0** |
| 001 | 002 | 002 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | | 0 | | 0 | 0 |
| 001 | 004 | 002 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | | 0 | | 0 | 0 |
| 001 | 006 | 002 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | | 0 | | 0 | 0 |
| **002** | **003** | **105** | **1** | **1** | **1** | **1** | **1** | **1** | **1** | | **1** | | **0** | **0** |
| **002** | **005** | **105** | **1** | **1** | **0** | **0** | **1** | **1** | **0** | | **0** | | **0** | **0** |
| **002** | **007** | **105** | **1** | **1** | **0** | **0** | **0** | **0** | **0** | | **0** | | **0** | **0** |
| 002 | 003 | 002 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | | 0 | | 0 | 0 |
| 002 | 005 | 002 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | | 0 | | 0 | 0 |
| 002 | 007 | 002 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | | 0 | | 0 | 0 |

# contract Cw1 is implemented by the green rows, contract Cw2 is implemented by the blue rows

|  |  |  |
| --- | --- | --- |
| **messageType\_T** | (\*) |  |
| messageType\_id | messageType\_title | messageType\_info |
| 1 | urgent | message that needs an alert sent to receiver email |
| 2 | normal | no need to send email alert |

|  |  |  |
| --- | --- | --- |
| **messageStatus\_T** | (\*) |  |
| messageStatus\_id | messageStatus\_title | messageStatus\_info |
| 1 | new | new message that is not read yet |
| 2 | read | message that has been read |
| 3 | deleted | not visible any more |

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **message\_T** | (\*) |  |  |  |  |  |  |  |
| message\_id | message\_type | message\_status | task\_id | message \_sender | message \_receiver | message\_subject | message\_contents | message\_datalink | message\_created |
| 000001 | 2 | 2 | 0001 | 002 | 004 | hi | let’s have first meeting on Monday at 10am | < none or id of data item> | 2019-01-19 05:10:28 |
| 000002 | 1 | 2 | 0002 | 007 | 003 | start | Can you review this form? | 0105 | 2019-01-19 09:13:07 |
| 000004 | 1 | 1 | 0002 | 003 | 007 | Re: start | Done, approved. | 0105 | 2019-01-19 19:13:07 |

|  |  |  |
| --- | --- | --- |
| **taskType\_T** |  |  |
| taskType\_id | taskType\_title | taskType\_info |
| 1 | urgent | task with high priority that needs to get done asap |
| 2 | normal | task with normal priority |

|  |  |  |
| --- | --- | --- |
| **taskStatus\_T** |  |  |
| taskStatus\_id | taskStatus\_title | taskStatus\_info |
| 1 | Approved | task that has been done, can’t be changed any more |
| 2 | In Progress | the task that is being worked on (at least one user is still in progress) |
| 3 | Rejected | task that has at least one user rejected |
| 4 | Deleted | not visible any more |

|  |  |  |
| --- | --- | --- |
| **templateStatus\_T** |  |  |
| templateStatus\_id | templateStatus\_title | templateStatus\_info |
| 1 | ready | template is ready to be used |
| 2 | not-ready | the template that is not ready or should not be used now |
| 3 | deleted | not visible any more |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **taskTemplate\_T** |  |  |  |  |  |
| taskTemplate\_id | policy\_id | task\_type | task\_title | task\_instructions | templateStatus\_id |
| 001 | 02 | 2 | ABC company report | The manager uploads part A, the staff adds and upload part B and customer can view (download) it. | 1 |
| 002 | 03 | 1 | TV repair Trip | Manager creates/selects a travel form. Staff fill-in Traveler’s name and Travel Mode. Manager selects the Total Cost. Customer views and signs with full name. | 1 |

|  |  |  |
| --- | --- | --- |
| **taskTemplateDetails\_T** |  |  |
| taskTemplate\_id | userRole\_id | dataType\_id |
| 001 | 2 | 4 |
| 001 | 3 | 4 |
| 001 | 4 | 4 |
| 002 | 2 | 5 |
| 002 | 3 | 5 |
| 002 | 4 | 5 |

|  |  |  |  |
| --- | --- | --- | --- |
| **jobTemplate\_T** |  |  |  |
| jobTemplate\_id | job\_title | job\_instructions | templateStatus\_id |
| 001 | TV service | The TV evaluation report is made by a technical manager and a staff and viewed by a customer. Then, a travel plan is put together by a service manager and a staff and is signed by the customer with a broken TV. | 1 |

|  |  |  |
| --- | --- | --- |
| **jobTemplateDetails\_T** |  |  |
| jobTemplate\_id | taskTemplate\_id | job\_taskOrder |
| 001 | 001 | 1 |
| 001 | 002 | 2 |

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **task\_T** |  |  |  |  |  |  |  |  |
| task\_id | task\_status | task\_title | task\_instructions | task\_type | contract\_id | task\_owner | task\_deadline | task\_created | |
| 000001 | 1 | ABC company report | The manager uploads part A, the staff adds and upload part B and customer can view (download) it. | 2 | 02 | 002 | 2019-01-19 05:10:28 | 2019-01-19 05:10:28 | |
| 000002 | 1 | TV repair Trip | Manager creates/selects a travel form. Staff fill-in Traveler’s name and Travel Mode. Manager selects the Total Cost. Customer views and signs with full name. | 1 | 03 | 003 | 2019-01-19 09:13:07 | 2019-01-19 09:13:07 | |

# task\_status will have the same data type as taskStatus\_id

# a task is done/completed when task\_status =1 and when all of its parts (taskPart\_status =1 in **taskDetails\_T** ) by all involved users are done.

|  |  |  |  |
| --- | --- | --- | --- |
| **taskDetails\_T** |  |  |  |
| task\_id | user\_id | data\_id | taskPart\_status |
| 000001 | 002 | 0104 | 1 |
| 000001 | 004 | 0104 | 1 |
| 000001 | 006 | 0104 | 1 |
| 000002 | 003 | 0105 | 1 |
| 000002 | 005 | 0105 | 1 |
| 000002 | 007 | 0105 | 2 |

# if every user is done with every data (all taskPart\_status =1, with the same task\_id) then usually the task is done/completed which leads to setting task\_status =1 in task\_T table;

|  |  |  |
| --- | --- | --- |
| **jobType\_T** |  |  |
| jobType\_id | jobType\_title | jobType\_info |
| 1 | urgent | job with high priority that needs to get done asap |
| 2 | normal | job with normal priority |

|  |  |  |
| --- | --- | --- |
| **jobStatus\_T** |  |  |
| jobStatus\_id | jobStatus\_title | jobStatus\_info |
| 1 | Done | job that has all of its tasks done/approved |
| 2 | In Progress | job that has at least one task NOT done/approved |
| 3 | Deleted | not visible any more |

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **job\_T** |  |  |  |  |  |  |  |
| job\_id | jobStatus\_id | job\_title | job\_instructions | jobType\_id | job\_deadline | job\_created | job\_owner |
| 000001 | 0 | TV service | The TV evaluation report is made by a technical manager and a staff and viewed by a customer. Then, a travel plan is put together by a service manager and a staff and is signed by the customer with a broken TV. | 1 | 2019-01-19 05:10:28 | 2019-01-19 05:10:28 | 002 |

# a app is done/completed when app\_status =1

|  |  |  |
| --- | --- | --- |
| **jobDetails\_T** |  |  |
| job\_id | task\_id | job\_taskOrder |
| 000001 | 000001 | 1 |
| 000001 | 000002 | 2 |

# if every task is done (all task\_status =1, with the same job\_id) then usually the job is done/completed which leads to setting job\_status =1 in job\_T table;

|  |  |  |  |
| --- | --- | --- | --- |
| **formElementType\_T** |  |  |  |
| formElementType\_id | formElementType \_title | formElementType \_zise | formElementType \_codeLocation |
| 1 | label | 1 | /form/Elements/formTemplateElement\_01.txt |
| 2 | textbox | 1 | /form/ Elements/formTemplateElement\_02.txt |
| 3 | radio | 2 | /form/ Elements/formTemplateElement\_03.txt |
| 4 | check | 2 | /form/ Elements/formTemplateElement\_04.txt |
| 5 | Select/dropdown | 2 | /form/ Elements/formTemplateElement\_05.txt |
| 6 | button | 1 | /form/ Elements/formTemplateElement\_06.txt |
| 7 | radio | 3 | … |
| 8 | check | 3 | … |
| 9 | Select/dropdown | 3 | /form/ Elements/formTemplateElement\_09.txt |
| 10 | radio | 4 | /form/ Elements/formTemplateElement\_10.txt |
| 11 | check | 4 | … |
| 12 | Select/dropdown | 4 | … |

# files can be in TEXT/JSON/PHP format => up to you

# can build GENERIC codes in the examples here: https://www.w3schools.com/html/html\_form\_elements.asp

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **formTemplate\_T** |  |  |  |  |  |  |
| formTemplate\_id | templateStatus\_id | formTemplate\_title | formTemplate\_instructions | formTemplate \_location | formTemplate \_owner | formTemplate \_created |
| 001 | 1 | Report form | Has a label with a link to the file and two button: download and upload | /form/temp/formT\_001.php | 001 | 2019-01-19 05:10:28 |
| 002 | 1 | Travel form | Has 5 elements:   * Element #1: label for Form title; * Element #2: textbox for Traveler’s name; * Element #3: radio buttons (of 2) for Travel Mode; * Element #4: select/dropdowns (of 2) for the Total Cost. * Element #5: textbox for for the Customer to sign | /form/temp/formT\_002.php | 001 | 2019-01-19 05:10:28 |

# files can be in TEXT/JSON/PHP format => up to you

|  |  |  |  |
| --- | --- | --- | --- |
| **formTemplateDetails\_T** |  |  |  |
| formTemplate\_id | formElementType\_id | formTemplate \_ElementOrder | formTemplate \_contents |
| 001 | 1 | 1 | /form/temp/formT\_001\_p1.txt |
| 001 | 6 | 2 | /form/ temp/formT\_001\_p2.txt |
| 001 | 6 | 3 | /form/ temp/formT\_001\_p3.txt |
| 002 | 1 | 1 | /form/temp/formT\_001\_p1.txt |
| 002 | 2 | 2 | /form/ temp/formT\_001\_p2.txt |
| 002 | 3 | 3 | /form/ temp/formT\_001\_p3.txt |
| 002 | 5 | 4 | /form/ temp/formT\_001\_p4.txt |
| 002 | 6 | 5 | /form/ temp/formT\_001\_p5.txt |
| 002 | 6 | 6 | /form/ temp/formT\_001\_p6.txt |

# files can be in TEXT/JSON/PHP format => up to you

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **form\_T** |  |  |  |  |  |  |  |  |
| form\_id | form\_status | form\_title | form\_instructions | form \_server | form\_modifier | form\_changed | form\_owner | form\_created |
| 0001 | 1 | report | Has a label with a link to the file and two button: download and upload | /form/form\_0001\_server.php | 002 | 2025-01-19 05:10:28 | 002 | 2019-01-19 05:10:28 |
| 0002 | 1 | travel | Has 5 elements:   * Element #1: label for Form title; * Element #2: textbox for Traveler’s name; * Element #3: radio buttons (of 2) for Travel Mode; * Element #4: select/dropdowns (of 2) for the Total Cost. * Element #5: textbox for for the Customer to sign | /form/form\_0002\_server.php | 003 | 2025-01-19 05:10:28 | 003 | 2019-01-19 05:10:28 |

# files can be in TEXT/JSON/PHP format => up to you

|  |  |  |  |
| --- | --- | --- | --- |
| **formDetails\_T** |  |  |  |
| form\_id | formElementType\_id | form \_ElementOrder | form \_contents |
| 001 | 1 | 1 | /form/temp/formT\_001\_p1.txt |
| 001 | 6 | 2 | /form/ temp/formT\_001\_p2.txt |
| 001 | 6 | 3 | /form/ temp/formT\_001\_p3.txt |
| 002 | 1 | 1 | /form/temp/formT\_001\_p1.txt |
| 002 | 2 | 2 | /form/ temp/formT\_001\_p2.txt |
| 002 | 3 | 3 | /form/ temp/formT\_001\_p3.txt |
| 002 | 5 | 4 | /form/ temp/formT\_001\_p4.txt |
| 002 | 6 | 5 | /form/ temp/formT\_001\_p5.txt |
| 002 | 6 | 6 | /form/ temp/formT\_001\_p6.txt |

# files can be in TEXT/JSON/PHP format => up to you

Form Elements/Components are store in files, for examples:

**GENERIC codes for ELEMENTS**

|  |  |
| --- | --- |
| /form/elements/formTemplateElement\_01.txt |  |
| <p>XXX<p> | 🡸 xxx is the GENERIC name of TEXTBOX variable |

|  |  |
| --- | --- |
| /form/elements/formTemplateElement\_02.txt |  |
| <label for="xxx">xxx\_label:</label><br> <input type="text" id="xxx" name="xxx"><br> | 🡸 xxx is the GENERIC name of TEXTBOX variable  xxx\_label is a GENERIC value in the STANDARD codes |

|  |  |
| --- | --- |
| /form/elements/formTemplateElement\_03.txt |  |
| 2  <label for="xxx"> xxx\_label</label>    <input type="radio" id="v1" name="xxx" value="v1">   <label for="v1">v1\_label</label><br>   <input type="radio" id="v2" name="xxx" value="v2">   <label for="v2">v2\_label</label><br> | 🡸 size or number of radio buttons  🡸 xxx is the GENERIC RADIO variable name  v1\_label, v2\_label, xxx\_label are GENERIC values |

|  |  |
| --- | --- |
| /form/elements/formTemplateElement\_04.txt |  |
| 2  <label> xxx\_label </label>  <input type="checkbox" id="x1" name="x1" value="v1">   <label for="x1"> x1\_label</label><br>   <input type="checkbox" id="x2" name="x2" value="v2">   <label for="x2"> x2\_label</label><br> | 🡸 size or number of CHECK BOXES  🡸 x1,x2 are the GENERIC CHECKBOXES variable name  x1\_label,x2\_label, xxx\_label are GENERIC values |

|  |  |
| --- | --- |
| /form/ Elements/formTemplateElement\_05.txt |  |
| 2  <label for="cars"> xxx\_label </label> <select id="XXX" name="XXX">   <option value="v1"> v1\_label </option>   <option value="v2"> v1\_label </option> </select> | 🡸 size or number of SLECT or DROPDOWN  🡸 xxx is the GENERIC variable name  v1\_label, v2\_label, xxx\_label are GENERIC values |

…

|  |  |
| --- | --- |
| /form/elements/formTemplateElement\_09.txt |  |
| 3  <label> xxx\_label </label>  <input type="checkbox" id="x1" name="x1" value="v1">   <label for="x1"> x1\_label</label><br>   <input type="checkbox" id="x2" name="x2" value="v2">   <label for="x2"> x2\_label</label><br>   <input type="checkbox" id="x3" name="x3" value="v3">   <label for="x3"> x3\_label</label><br> | 🡸 size or number of CHECK BOXES  🡸 x1,x2,x3 are the GENERIC CHECKBOXES variable name  x1\_label,x2\_label,x3\_label, xxx\_label are GENERIC values |

…

**SPECIFIC Codes for ELEMENTS in form TEMPLATES:**

|  |  |
| --- | --- |
| /form/ temp/formT\_001\_p02.txt |  |
| <label for="t1">**Traveler’s Name:</**label><br> <input type="text" id="t1" name="t1"><br> | 🡸 xxx = **t1**  xxx\_label = **Traveler’s Name** (for THIS SPECIFIC form template) |

|  |  |
| --- | --- |
| /form/ temp/formT\_001\_p04.txt |  |
| 2  <label for="tMode"> Travel Mode </label>    <input type="radio" id="10" name="tMode" value="10">   <label for="10">Car</label><br>   <input type="radio" id="20" name="tMode" value="20">   <label for="20">Plane</label><br> | 🡸 size or number of RADIO buttons  🡸 xxx = tMode  v1\_label = Car  v2\_label = Plane  label = Travel Mode |

|  |  |
| --- | --- |
| /form/ temp/formT\_003\_p19.txt |  |
| 3  <label> xxx\_label </label>  <input type="checkbox" id="x1" name="x1" value="v1">   <label for="x1"> x1\_label</label><br>   <input type="checkbox" id="x2" name="x2" value="v2">   <label for="x2"> x2\_label</label><br>   <input type="checkbox" id="x3" name="x3" value="v3">   <label for="x3"> x3\_label</label><br> | 🡸 size or number of CHECKBOXES  🡸 x1,x2,x3 are the GENERIC CHECKBOXES variable name  x1\_label,x2\_label, and x3\_label are GENERIC values |

**…**

|  |  |
| --- | --- |
| /form/ temp/formT\_003\_p19.txt |  |
| 3  <label> Select Your Vehicle Type:</label>  <input type="checkbox" id="vehicle1" name="vehicle1" value="Bike">   <label for="vehicle1"> I have a bike</label><br>   <input type="checkbox" id="vehicle2" name="vehicle2" value="Car">   <label for="vehicle2"> I have a car</label><br>   <input type="checkbox" id="vehicle3" name="vehicle3" value="Boat">   <label for="vehicle3"> I have a boat</label> | 🡸 size or number of CHECK BOXES  🡸 xx\_label = Select Your Vehicle Type:  x1=vehicle1; x2=vehicle2, x3=vehicle3  x1\_label=Bike, x2\_label=Car, and x3\_label=Boat  Will need to (can you?) extract the variable names and their values so that the PHP server code knows what and in which order to process. |

…

**2.2 List of MySQL commands to CREATE tables and INSERT data (in 2.1):**

<please put here MySQL commands needed to create and insert data in 2.1, you should also put these in a db.txt file so that you can easily recreate them later>

For example:

CREATE TABLE ` **userRole\_T**` (

` userRole\_id` int(3) NOT NULL AUTO\_INCREMENT,

` userRole\_title` varchar(50) NOT NULL,

` userRole\_info` varchar(255) NOT NULL,

PRIMARY KEY (`userRole\_id`)

) ENGINE=InnoDB DEFAULT CHARSET=latin1 AUTO\_INCREMENT=1 ;

…

**2.3 EERD for tables:**

<please insert the new diagram (similar to which you already have) for the new set of tables above, try to put them into two groups/areas (within the same diagram): user-related and policy/permission-related ones>

**(3). User ACTIONS/TRANSACTIONS**

# please put these (over 20 actions in sections as listed below) on the LEFT VERTICAL menu of your unit on your COS website

# the implementation of these are based on the following **BASIC operations** with a table, so you should start with it => make it version/folder **v1**)

**# VIEW = display all rows in a table**

**# CREATE = add a new row to a table**

**# DELETE = remove a row in a table**

**# EDIT = display fields of a row in a table in an input TEXTBOX and let user change and submit/write back to the table**

**3.1 List of ACTIONS/TRANSACTIONS**

< The following is what should be (for now) on the LEFT VERTICAL menu of your unit on your COS website, try to implement them (this will be version/folder **v2**) , have a php file for each, NAME the file this way: XX-NNN.php where XX is your group ID with lowercase letter and NNN is some short name for the given action, for examples: a1-userOneView.php, a1-userAllView.php, a1-policyOneView.php, a1-policyAllView,php … Below I put down some names for some cases for examples. Please FILL-IN the names for code files for all actions in your document>

< Later you may need to move these around for better look but if you already implement them then it won’t be a lot of work just to move them around>

=========================================================================================================================

**UNIT A**

USER (SELF, data=Profile)

* Register (codes in “a1-userRegister.php”, modify existing codes with the new tables, keep passwords un-hashed)
* Login (codes in “a1-userLogin.php”, modify existing codes with the new tables, keep passwords un-hashed)
* Logout (codes in “a1-userLogout.php”)
* View Profile (fields from this row in user\_T, codes in “a1-userOneView.php”)
* Edit Profile (a field in this row in user\_T, codes in “a1-userOneEdit.php”)

USER (OTHER, data=user\_T)

* View all users (in user\_T, codes in “a1-userAllView.php”)
* View users with filters (in user\_T, …)
* Add a new user (add a new row to user\_T, codes in “a1-userAdd.php”)
* Delete a new user (add a new row to user\_T, …)
* Edit a user Profile (fields in a row from user\_T, codes in “a1-userOneEdit.php”)
* Approve a user (change status field in this row in user\_T, …)
* Activate a user (change status field in this row in user\_T, …)
* Deactivate a user (change status field in this row in user\_T, …)

POLICY (SELF or OTHER, data=policy\_T and policyDetails\_T)

* View all policies (in policy\_T and policyDetails\_T )
* View policies with filters (in policy\_T)
* Add a new policy (add a new row to policy\_T and policyDetails\_T)
* Delete a new policy (add a new row to policy\_T and policyDetails\_T)
* Edit a policy Profile (fields in a row from policy\_T and policyDetails\_T)
* Approve a policy (change status field in this row in policy\_T)
* Activate a policy (change status field in this row in policy\_T)
* Deactivate a policy (change status field in this row in policy\_T)

CONTRACT (SELF or OTHER, data=contract\_T and contractDetails\_T)

* View all contracts (in contract\_T and contractDetails\_T )
* View contracts with filters (in contract\_T)
* Add a new contract (add a new row to contract\_T and contractDetails\_T)
* Delete a new contract (add a new row to contract\_T and contractDetails\_T)
* Edit a contract Profile (fields in a row from contract\_T and contractDetails\_T)
* Approve a contract (change status field in this row in contract\_T)
* Activate a contract (change status field in this row in contract\_T)
* Deactivate a contract (change status field in this row in contract\_T)

=========================================================================================================================

**UNIT B**

**Task TEMPLATE**

* **View all task TEMPLATEs**
* **(Select and) Delete a task TEMPLATE [change its status to “Deleted”]**
* **Create a new task TEMPLATE**
* **(Select and) Edit a task TEMPLATE**
* **Deactivate a task TEMPLATE [change its status to “Not-ready”]**
* **Activate a task TEMPLATE [change its status to “Ready”]**

**Job TEMPLATE**

* **View all job TEMPLATEs**
* **(Select and )Delete a job TEMPLATE**
* **Create a new job TEMPLATE**
* **(Select and) Edit a job TEMPLATE**
* **Deactivate a job TEMPLATE**
* **Activate a job TEMPLATE**

**TASK**

* **View all tasks**
* **(Select and) Delete a task**
* **Create a new task**
* **(Select and) Edit a task**
* **Pause a task**
* **Resume a task**
* **Finish a task [check if all parts are complete then change its status to “done”, lock all data]**

**JOB**

* **View all jobs**
* **(Select and ) Delete a job**
* **Create a new job**
* **(Select and) Edit a job**
* **Pause a job [change its status to “Paused”]**
* **Resume a job [change its status to “Progress”]**
* **Finish a job [check if all parts are complete then change its status to “done” , lock all data]**

=========================================================================================================================

**UNIT C**

DATA

* View all data belonging to this user (in data\_T, codes in “a1-dataAllView.php”)
* View data belonging to this user with filters (in data\_T, …)
* Add a new data (add a new row to data\_T, codes in “a1-dataAdd.php”)
* Delete a data (delete a row in data\_T, …)
* Edit a data belonging to this user (codes in “a1-dataOneEdit.php”)
* Approve a data (change status field in this row in data\_T, codes in “a1-dataApprove.php”)
* Reject a data (change status field in this row in data\_T, …)
* Activate a data (change status field in this row in data\_T, …)
* Deactivate a data (change status field in this row in data\_T, …)
* Approve a data (change status field in this row in data\_T, codes in “a1-dataApprove.php”)
* Reject/disapprove a data (change status field in this row in data\_T, …)
* UnLock a data (change status field in this row in data\_T, codes in “a1-dataUnlock.php”)
* Lock a data (change status field in this row in data\_T, …)
* View all data (in data\_T, codes in “a1-dataAllView.php”)
* View all data with filters (in data\_T, …)
* Select one and Edit a data (fields in a row from data\_T, codes in “a1-dataAllEdit.php”)
* Approve a data (change status field in this row in data\_T, codes in “a1-dataApprove.php”)
* Reject a data (change status field in this row in data\_T, …)
* Activate a data (change status field in this row in data\_T, …)
* Deactivate a data (change status field in this row in data\_T, …)
* UnLock a data (change status field in this row in data\_T, codes in “a1-dataUnlock.php”)
* Lock a data (change status field in this row in data\_T, …)

POLICY (SELF or OTHER, data=policy\_T and policyDetails\_T)

* View all policies (in policy\_T and policyDetails\_T )
* View policies with filters (in policy\_T)
* Add a new policy (add a new row to policy\_T and policyDetails\_T)
* Delete a new policy (add a new row to policy\_T and policyDetails\_T)
* Edit a policy Profile (fields in a row from policy\_T and policyDetails\_T)
* Approve a policy (change status field in this row in policy\_T)
* Activate a policy (change status field in this row in policy\_T)
* Deactivate a policy (change status field in this row in policy\_T)

CONTRACT (SELF or OTHER, data=contract\_T and contractDetails\_T)

* View all contracts (in contract\_T and contractDetails\_T )
* View contracts with filters (in contract\_T)
* Add a new contract (add a new row to contract\_T and contractDetails\_T)
* Delete a new contract (add a new row to contract\_T and contractDetails\_T)
* Edit a contract Profile (fields in a row from contract\_T and contractDetails\_T)
* Approve a contract (change status field in this row in contract\_T)
* Activate a contract (change status field in this row in contract\_T)
* Deactivate a contract (change status field in this row in contract\_T)

=========================================================================================================================

**UNIT D**

**Element TEMPLATE**

* **View all Element TEMPLATEs**
* **(Select and) Delete an Element TEMPLATE [change its status to “Deleted”]**
* **Create a new Element TEMPLATE**
* **(Select and) Edit an Element TEMPLATE**
* **Deactivate an Element TEMPLATE [change its status to “Not-ready”]**
* **Activate an Element TEMPLATE [change its status to “Ready”]**

**Form TEMPLATE**

* **View all form TEMPLATEs**
* **(Select and) Delete a form TEMPLATE**
* **Create a new form TEMPLATE**
* **(Select and) Edit a form TEMPLATE**
* **Deactivate a form TEMPLATE**
* **Activate a form TEMPLATE**

**ELEMENT**

* **View all Element**
* **(Select and) Delete a Element**
* **Create a new Element**
* **(Select and) Edit a Element**
* **Deactivate an Element [change its status to “Not-ready”]**
* **Activate an Element [change its status to “Ready”]**

**FORM**

* **View all Forms**
* **(Select and ) Delete a Form**
* **Create a new Form**
* **(Select and) Edit a Form**
* **Deactivate a form**
* **Activate a form**

=========================================================================================================================

**UNIT E**

**MESSAGES**

* **View all message in the system (in message\_T, codes in “a1-messageAllView.php”) sort by time,…**
* **View all message in the system with filters (in message\_T, …) NEW, IN(receiver=this user), OUT(sender=this user),**
* **View all message belonging to this user (in message\_T, codes in “a1-messageAllView.php”), sort by time,…**
* **View message belonging to this user with filters (in message\_T, …) NEW, IN(receiver=this user), OUT(sender=this user),**
* **Mark as read: after a new message (status=1/new) is viewed by the receiver (view by other users will not be counted) the status of given message should be changed to 2/read**
* **Delete a message (delete a row in message\_T, …) must first check (in contractDetails\_T) if the given user has permission to delete messages;**
* **Compose/Add(send) a new message (add a new row to message\_T, codes in “a1-messageAdd.php”): must first check (in contractDetails\_T) if the given user has permission to send messages; if so must have a form for this user to specify receiver, subject, contents, urgent or not, if it’s urgent then also needs to send an email (using user\_T to get the email of the receiver) to alert the receiver.**
* **Reply to a message (similar to Compose/Add)**
* **Forward a message (similar to Compose/Add)**

=========================================================================================================================

**3.2 PROTOCOLS of ACTIONS/TRANSACTIONS**

<please put here a diagram (final, **not** step-by-step) for each action/transaction, at this point it doesn’t need to be fancy but editable so that you can go back and change if necessary, the more details you try to put in the diagram the more you’ll understand about what you need in your codes, please do this while coding, go back and change if necessary>