legacy tasking system

Revision History

Date	Version	Description	Author
2017-02-14	0.01	Initial Draft	Shahin Sheidaei
2017-02-16	0.02	Initial Draft	Shahin Sheidaei

9.

Database structure

1. Introduction

In this project, we will implement a legacy tasking system that features dependencies. Each task will have an ID, title, status, and parent task ID. A task's status is IN PROGRESS until it is marked as DONE. While any task can be marked as DONE, a "parent" task (one with dependencies) must be marked as COMPLETE automatically when all of its dependencies (and sub-dependencies) are likewise marked as COMPLETE. A task is only considered COMPLETE when it's marked as DONE and either has no dependencies or all of its dependencies are also COMPLETE. Any individual task may have any number of dependencies but should never result in a circular dependency. For example, if Task A depends on Task B, then Task B cannot also depend on Task A.

1.1 Project's requirements

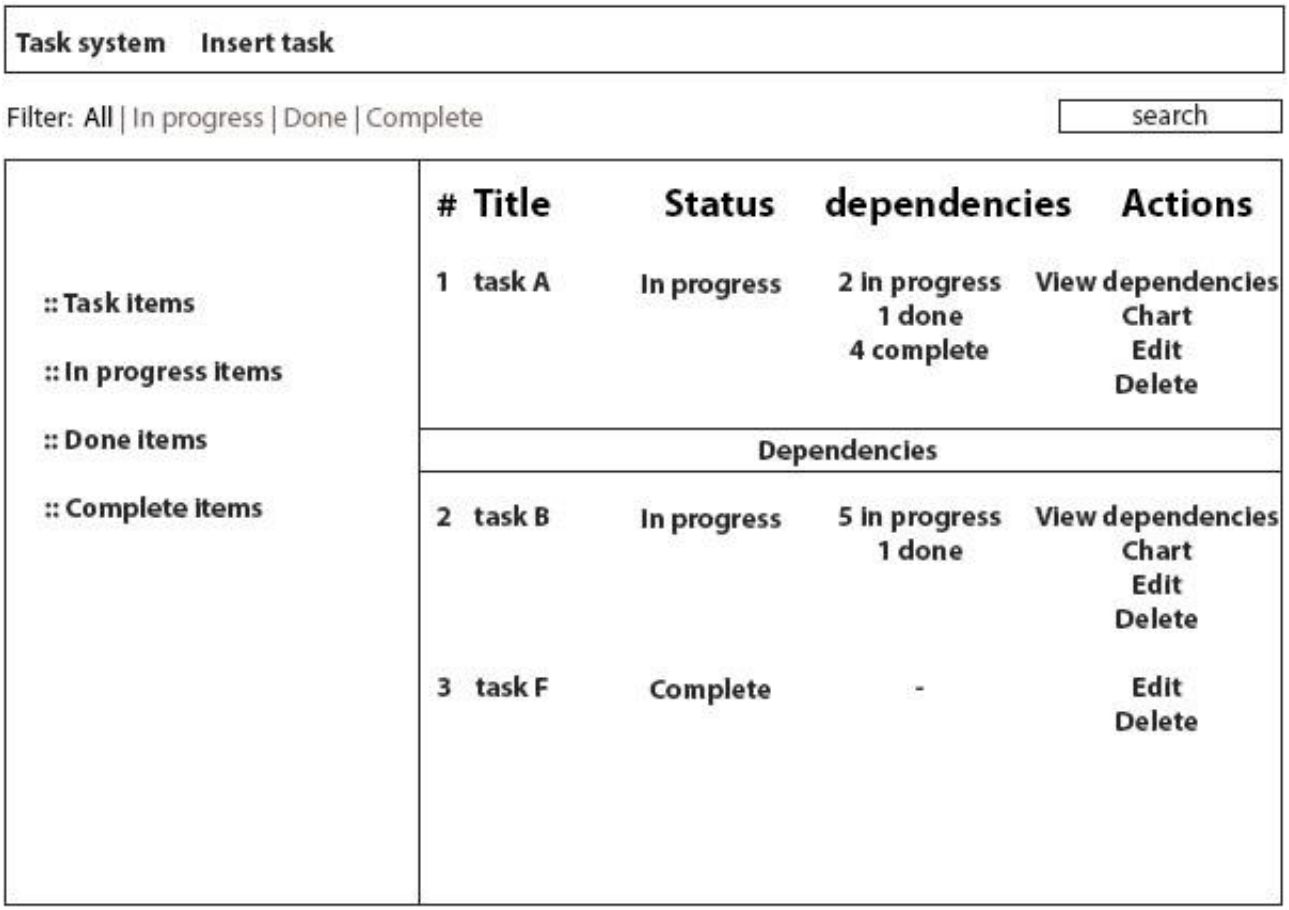
This web application implements the above logic with following requirements.
Complete as many requirements possible with the highest priority being those at the top of this list:

- Create a task listing page that shows a flat list of individual tasks and:
  - Each task's ID, description, and IN PROGRESS / DONE / COMPLETE status,
  - A status checkbox for each task to mark/unmark it has DONE when clicked.
- The task listing page should feature a status filter (IN PROGRESS, DONE, COMPLETE).
- Create a task creation form that takes the following inputs:
  - Task name (required),
  - Parent task ID (optional).
- Check for and prevent circular dependencies when creating a task that specifies a parent.
- Upgrade the task listing page so that parent tasks also show:
  - The total number of dependencies,
  - The number of dependencies marked as DONE,
  - The number of dependencies marked as COMPLETE.
- Upgrade the task listing page so that:
  - Marking a task as DONE will also check the status of all dependencies. When all dependencies are COMPLETE, mark the task as COMPLETE (instead of DONE).
  - Marking a task as IN PROGRESS (by clearing the status checkbox) should update its parent task (if it has one) so that the parent's status changes from COMPLETE to DONE. A parent task must not revert to IN PROGRESS from DONE or COMPLETE.
  - Repeat these two processes on the task's parents (if any) until the status change has propagated all the way to the top of the hierarchy.
- Upgrade the task listing page from a flat list to a nested hierarchal list. That is, all of the dependencies for a task should appear in a separate list within the parent task.
- Upgrade the task listing page to allow tasks to be edited:
  - Allow a task's name to be changed.
  - Allow a task's parent task to be changed. Doing this must trigger a status change
  - propagation behaviour as described above.
- Optional / bonus features:
  - The task listing page should use pagination to show only 20 tasks at a time. This applies to the root level as well as for dependencies within a task.
  - Real time update of task listing page when changes are made by others.

- Use AJAX wherever possible to reduce or eliminate page loads.

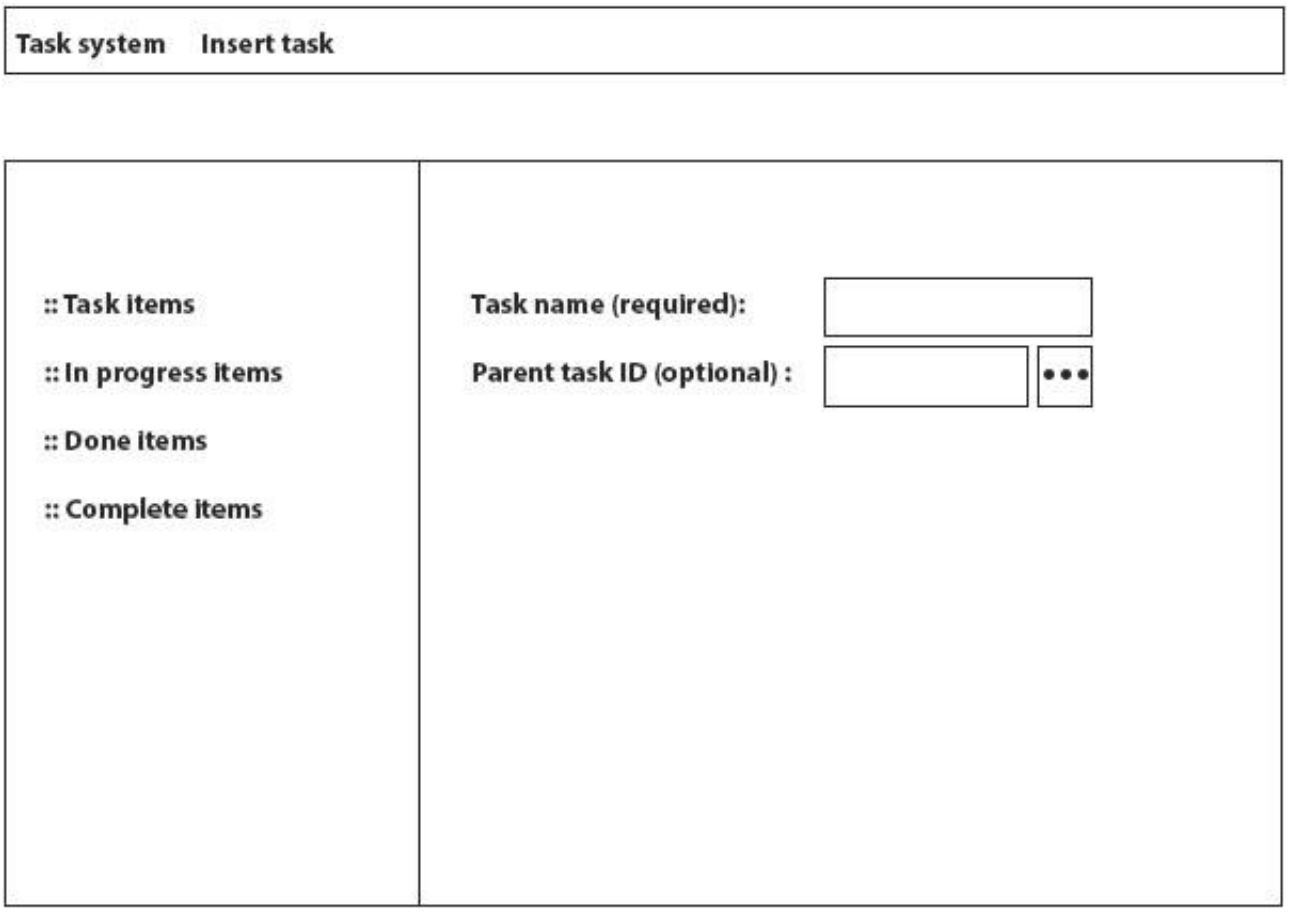
1.2 Tasks list

This page displays a flat list of individual tasks. User can see dependencies by click on 'view dependencies' icon, it shows a list of dependencies that are related to previous record. There are some other features like search and filter the list.



1- Tasks list

1.3 Insert task



2- Insert task

In this page user able to insert a task. Task name is required but parent task can be empty, by the way if users want to enter a dependency, they can enter parent id or click "…" button and choose one parent from list in popup window.

1.4 Definitions and acronyms

1.4.1 Definitions

Keyword	Definitions
DONE status	The DONE status is specific to a single task by itself
COMPLETE status	the COMPLETE status reflects the statuses of all dependencies.

circular dependency	A circular dependency is a chain of dependencies that loops back on itself.

1.4.2 Acronyms and abbreviations

Acronym or abbreviation	**Definitions**

1.5 References

2. Organization

2.1 Project Manager

Streamline Studios project owner

Shahin Sheidaei Project manager

2.2 Project Group

Name	Responsibility
Streamline Studios	Idea processor, Project owner, Documentation, Analysis, DB Design, Testing
Shahin Sheidaei	Project manager, Documentation, Analysis, Implementation, Designing, DB Design, Integration, Testing

2.3 Steering Group

Shahin sheidaei

2.4 Customer

2.5 Others

3. Milestones

Id	Milestone Description	Responsible Dept./Initials	Finished week				Met r	Rem
			Plan	Forecast		Actual		
				Week	+/-			
M-001	Project Description & Plan							
M-002	Requirement Definition							
M-004	Project Design							

M-005	Revised Project Plan							
M-006	Project Status Presentation							
M-007	Final Presentation & delivery							

3.1 Remarks

Remark Id	Description

4. Project Results

4.1 Requirements

4.1.1 Requirement Compliance Matrix

Id	Requirement Description	completed	Rem

Completed: Yes (completely implemented)

No (not implemented at all)

Partially (partially implemented, more description under Remarks subsection)

Unknown (completion status not known)

Dropped (requirement was dropped during the course of the project)

4.1.2 Requirements Compliance Summary

Total number of requirements	
Number of requirements implemented	
Requirements partially fulfilled	
Requirements not fulfilled	
Requirements dropped	

4.1.3 Remarks

Remark Id	**Description**

4.2 Work Products and Deliverables

To	Output	Planned week	Promised week	Late +/-	Delivered week	Rem
Shahin Sheidaei	Project Description & Plan					
Shahin Sheidaei	Requirement Definition					
Shahin Sheidaei	Project Design					
Shahin Sheidaei	Revised Project Plan					
Shahin Sheidaei	Project Status Presentation					
Shahin Sheidaei	Final Presentation & delivery					

4.2.1 Remarks

Remark Id	Description

5. Project Experiences

5.1 Positive Experiences

5.2 Improvement Possibilities

We have experience that if we will organize our resource according to requirements then we can make project more successful.

6. Financials

6.1 Project Cost Summary

Planned Cost	
Actual Cost	

6.2 Work per Member (Hour)

Member	W01											Total
Shahin Sheidaei												
Total												

7. Feasibility

The gold for feasibility is that check for is possible to do the features that describe at introduce system. Is one of the most important parts of doing project. (Definition and classification issues, opportunities and guidelines)

8. Analyse

8.1 System forms

8.1.1 Tasks

Through this form we able to insert a task.

Task name (required): … Parent task ID (optional): …

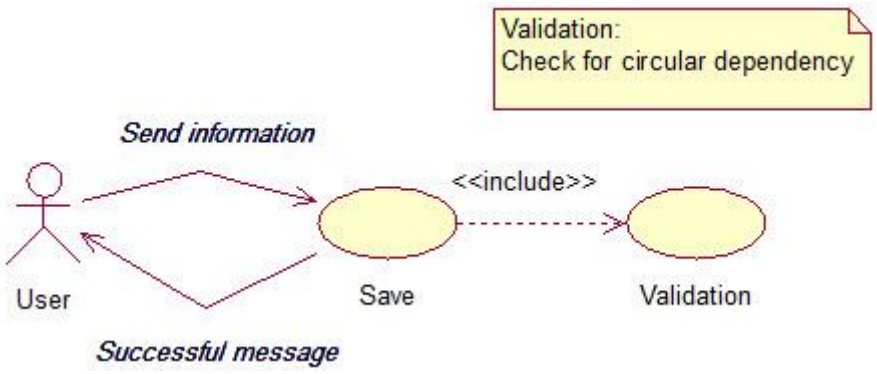
Fields

- Task name
  This field is a text field that gets task title.
- Parent task ID
  This field is a text field that gets parent task ID.

#	Title	Name	Type	Mandatory	Default value	Comment
1	task name	title	text	*		
2	Parent task ID	parent_id	number			

8.2 Use Case and their diagram (best and general)

8.2.1 Insert task - best

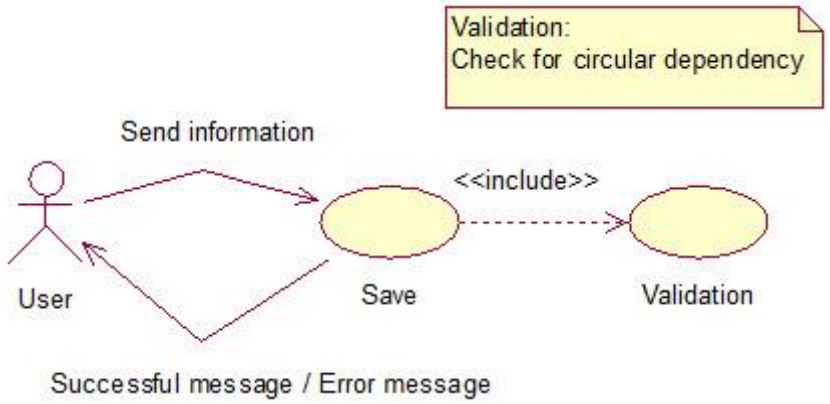


3- Use Case - best – insert task

8.2.1.1 Scenario (best)

- Fill the form
- Send it for validation and save into DB
- Give success message
- Logout from system

8.2.2 Insert task - general

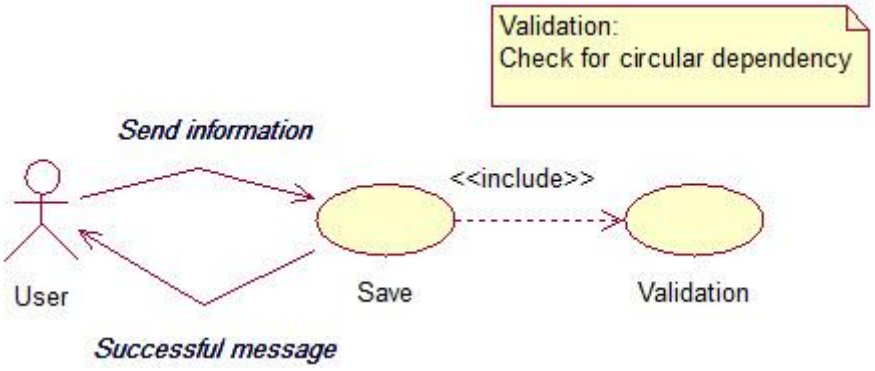


4- Use Case - general – insert task

8.2.2.1 Scenario (general)

- Fill the form
- Send it for validation and save into DB
- Check your information again if there is an error message
- Send your form
- Give success message
- Logout from system

8.2.3 Edit task - best

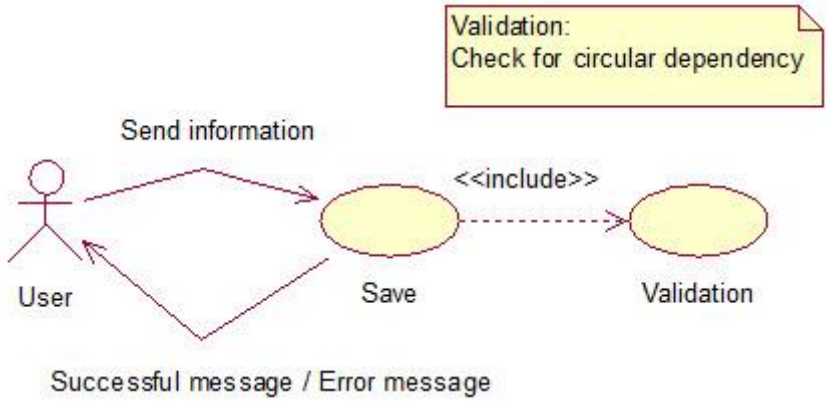


5- Use Case - best – edit task

8.2.3.1 Scenario (best)

- Fill the form
- Send it for validation and save into DB
- Give success message
- Logout from system

8.2.4 Edit task - general

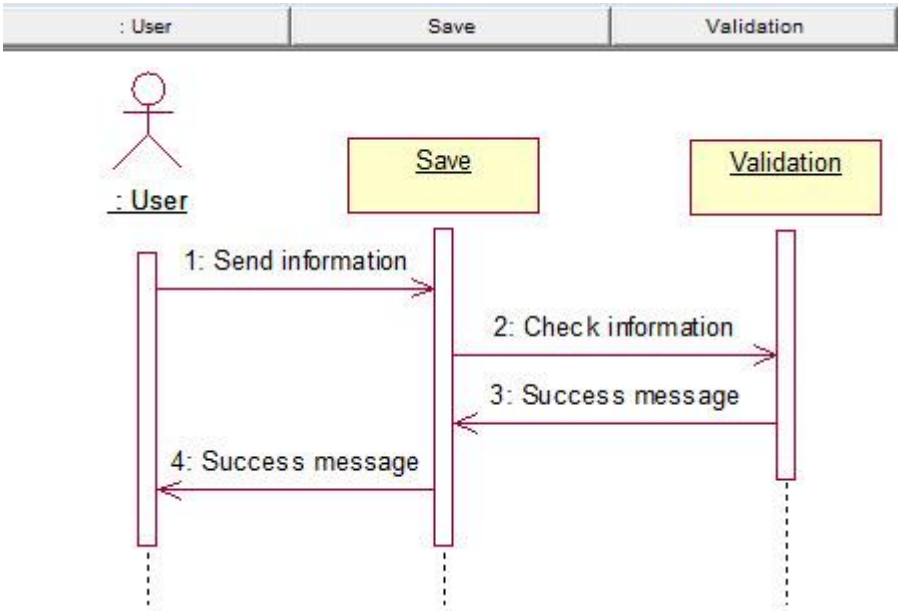


6- Use Case - best – edit task

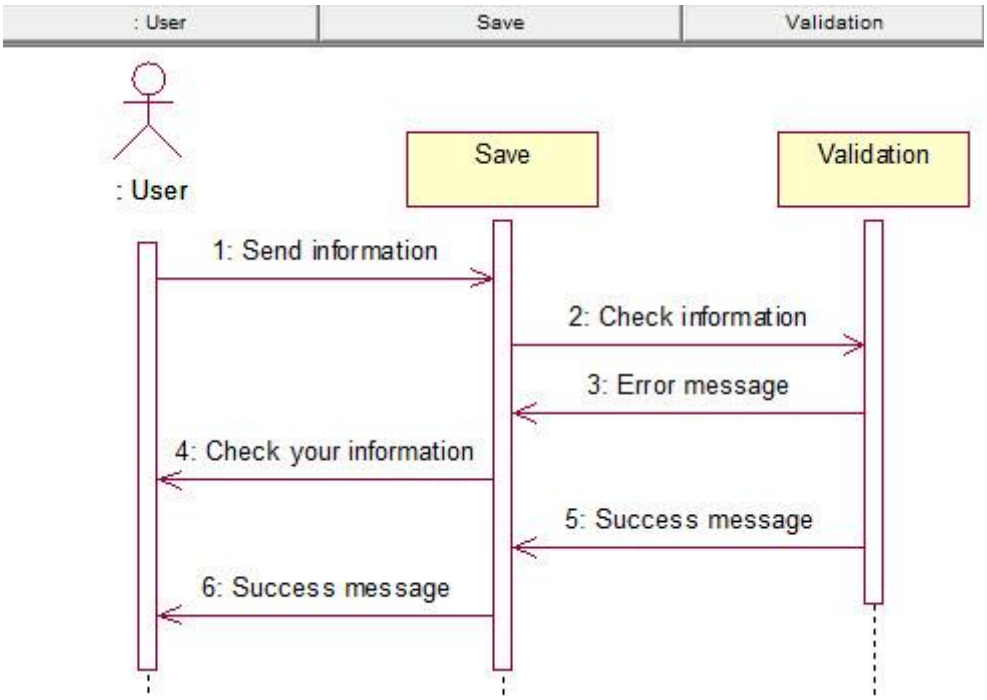
8.2.4.1 Scenario (general)

- Fill the form
- Send it for validation and save into DB
- Check your information again if there is an error message
- Send your form
- Give success message
- Logout from system

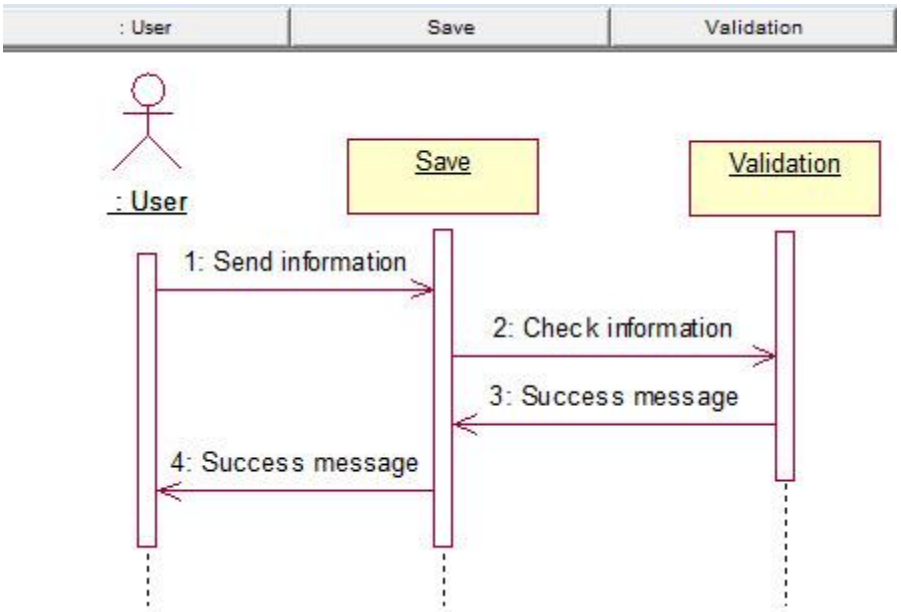
8.3 Sequence diagram (best and general)



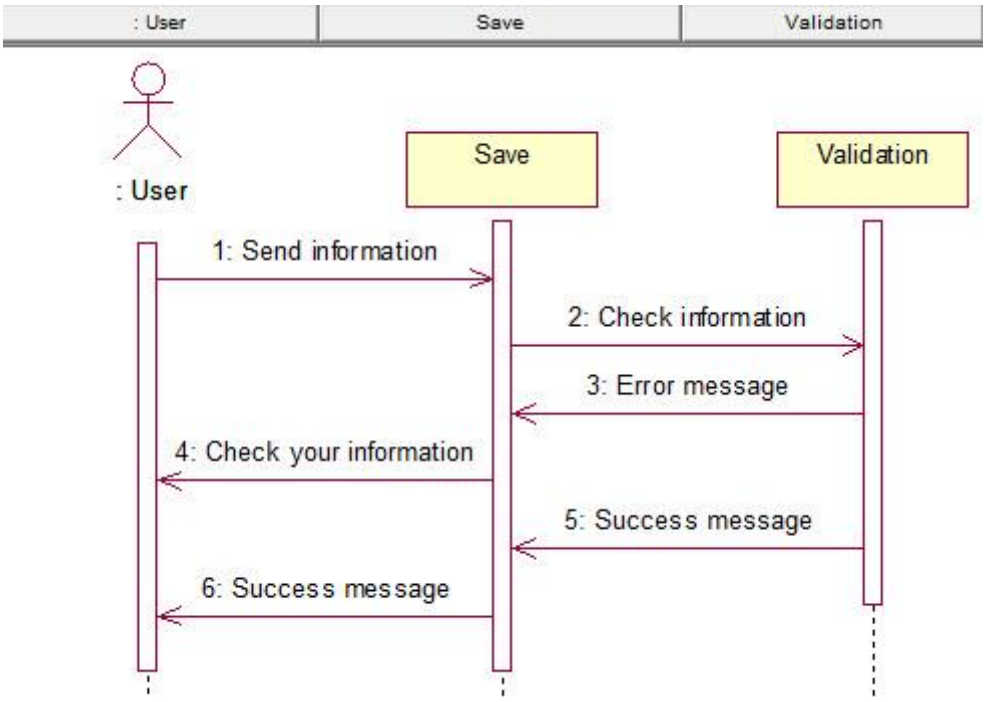
7- sequence – best - insert task



8- sequence – general - insert task



9- sequence – best - edit task



10- sequence – general - edit task

8.4 Class diagram

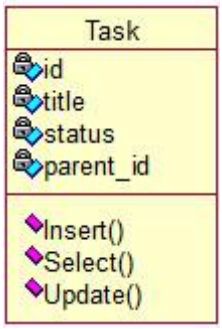


Figure 11- Class diagram

9. Database structure

Below is a description of the Task table's database structure.

Task Table

```
CREATE TABLE IF NOT EXISTS `tasks` (
    `id` int(11) NOT NULL AUTO_INCREMENT,
    `title` varchar(255) NOT NULL,
    `status` tinyint(1) NOT NULL DEFAULT '0',
```

```
    `parent_id` int(11) NOT NULL DEFAULT '0',
    PRIMARY KEY (`id`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8 AUTO_INCREMENT=1;
```

#	Title	Name	Type	Mandatory	Default value	Unique	Comment
1	id	id	int	*		*	
2	title	title	varchar				
3	status	status	tinyint		0		0 = IN PROGRESS 1 = DONE 2 = COMPLETE
3	parent task ID	parent_id	int		0		