**WIL Report Management System**

Software design

­­­

By

**Ms. Phinthip Samutloiwon 552115050**

**Mr. Veerapat In-ongkarn 562115055**

Department of Software Engineering

College of Arts, Media and Technology

Chiang Mai University

Project Advisor

**Dr. Prompong Sugunnasil**

Document History

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Document Name | Version | Status | | Date | Viewable | Reviewer | Responsible |
| Documents | | | | | | | |
| WRMS- SDD\_V.0.1.docx | * Add Chapter I   + Purpose   + Scope   + Acronym * Add Chapter II   + System architecture * Add Chapter III   + Class diagram   + Class diagram description   + Sequence diagram * Add Chapter IV   + UI design | | Draft | 18-02-2017 | PS,VI, PSU | PS,VI, PSU | PS,VI, PSU |
| WRMS- SDD\_V.0.2.docx | * Edit Chapter I   + Purpose   + Scope   + Acronym * Edit Chapter II * Edit Chapter III   + Class diagram   + Class diagram description   + Sequence diagram * Edit Chapter IV   + UI design | | Draft | 13-03-2017 | PS,VI, PSU | PS,VI, PSU | PS,VI, PSU |
| WRMS- SDD\_V.0.3.docx | * Edit Chapter III   + Class diagram   + Class diagram description   + Sequence diagram | | Draft | 14-03-2017 | PS,VI, PSU | PS,VI, PSU | PS,VI, PSU |
| WRMS- SDD\_V.0.4.docx | Release WRMS software design document | | Release | 09-05-2017 | PS,VI, PSU | PS,VI, PSU | PS,VI, PSU |

\*PS = Phinthip Samutloiwon  
\*VI = Veerapat In-ongkarn  
\*PSU= Prompong Sugunnasil

Table of Contents

[Chapter I | Introduction 6](#_Toc482052367)

[1.1 Purpose 6](#_Toc482052368)

[1.2 Scope 6](#_Toc482052369)

[1.3 Acronyms and Definition 6](#_Toc482052370)

[Chapter II | Class diagram 7](#_Toc482052371)

[2.1 System class diagram 7](#_Toc482052372)

[2.2 Class diagram description 8](#_Toc482052373)

[CD-01: Users 8](#_Toc482052374)

[CD-02: Project 9](#_Toc482052375)

[CD-03: Task 9](#_Toc482052376)

[CD-04: TaskLog 10](#_Toc482052377)

[CD-05: Comment 10](#_Toc482052378)

[CD-06: Notify 11](#_Toc482052379)

[CD-07: UserService 11](#_Toc482052380)

[CD-08: ProjectService 13](#_Toc482052381)

[CD-09: TaskService 14](#_Toc482052382)

[CD-10: CommentService 15](#_Toc482052383)

[CD-11: UserController 16](#_Toc482052384)

[CD-12: AuthController 17](#_Toc482052385)

[CD-13: ProjectController 18](#_Toc482052386)

[CD-14: TaskController 19](#_Toc482052387)

[CD-15: CommentController 21](#_Toc482052388)

[2.3 Sequence diagram 22](#_Toc482052389)

[2.3.1 Feature 1 22](#_Toc482052390)

[2.3.2 Feature 2 27](#_Toc482052391)

[2.3.3 Feature 3 32](#_Toc482052392)

[2.3.4 Feature 4 33](#_Toc482052393)

[2.3.5 Feature 5 34](#_Toc482052394)

[Chapter III | ER diagram 35](#_Toc482052395)

[3.1 Users table 35](#_Toc482052396)

[3.2 student\_has\_users table 35](#_Toc482052397)

[3.3 projects table 36](#_Toc482052398)

[3.4 task table 36](#_Toc482052399)

[3.5 task\_types table 36](#_Toc482052400)

[3.6 task\_logs table 37](#_Toc482052401)

[3.7 comments table 37](#_Toc482052402)

[3.8 notifies table 38](#_Toc482052403)

[3.9 users\_has\_notifies table 38](#_Toc482052404)

[Chapter V| UI Design 39](#_Toc482052405)

**Table of Figures**

[Figure 1 Class diagram of WRMS 7](#_Toc482062735)

[Figure 2 SD-01 Registration 22](#_Toc482062736)

[Figure 3 SD-02 Edit profile 23](#_Toc482062737)

[Figure 4 SD-03 Login 24](#_Toc482062738)

[Figure 5 SD-04 Logout 25](#_Toc482062739)

[Figure 6 SD-05 Add code 25](#_Toc482062740)

[Figure 7 SD-06 View task 26](#_Toc482062741)

[Figure 8 SD-07 View Statistics 26](#_Toc482062742)

[Figure 9 SD-08 Create project 27](#_Toc482062743)

[Figure 10 SD-09 Edit project 27](#_Toc482062744)

[Figure 11 SD-10 Delete project 28](#_Toc482062745)

[Figure 12 SD-11 Create task 28](#_Toc482062746)

[Figure 13 SD-12 Edit task 29](#_Toc482062747)

[Figure 14 SD-13 Delete task 29](#_Toc482062748)

[Figure 15 SD-14 Change a status of a task 30](#_Toc482062749)

[Figure 16 SD-15 View Comment 31](#_Toc482062750)

[Figure 17 SD-16 Create comment 31](#_Toc482062751)

[Figure 18 SD-17 Delete comment 32](#_Toc482062752)

[Figure 19 SD-18 View activities in a project 32](#_Toc482062753)

[Figure 20 SD-19 View activity of all Students 33](#_Toc482062754)

[Figure 21 SD-20 Generate a weekly report 33](#_Toc482062755)

[Figure 22 SD-21 Print a weekly report 34](#_Toc482062756)

[Figure 23 SD-22 Receive web notification 34](#_Toc482062757)

[Figure 24 SD-23 Receive Email notification 35](#_Toc482062758)

[Figure 25 Overview ER-diagram 36](#_Toc482062759)

[Figure 26 UI-01 Login 41](#_Toc482062760)

[Figure 27 UI-02 Profile 41](#_Toc482062761)

[Figure 28 UI-03 Edit profile 42](#_Toc482062762)

[Figure 29 UI-04 Student’s dashboard 42](#_Toc482062763)

[Figure 30 UI-05 Task page (List) 43](#_Toc482062764)

[Figure 31 UI-06 Task page (card) 43](#_Toc482062765)

[Figure 32 UI-07 Task page (activity) 44](#_Toc482062766)

[Figure 33 UI-08 Task detail page 44](#_Toc482062767)

[Figure 34 UI-09 Comment and notification 45](#_Toc482062768)

[Figure 35 UI-10 report page 45](#_Toc482062769)

[Figure 36 UI-11 Print the weekly report by browser 46](#_Toc482062770)

# **Chapter I | Introduction**

# **Purpose**

The purpose of this document is to plan the structure of the system follow from SRS. It will help developers to understand the overall system quicker and deeper detail. This document includes the software design of WIL Report management system which are System architecture, a system class diagram, detail description of class diagram, and sequence process of each use case.

# **Scope**

The scope of this document contain:

* Overall class diagram
* Detail of each class diagram including attribute and method
* Sequence diagram of each use case
* UI design

# **Acronyms and Definition**

|  |  |
| --- | --- |
| Acronym | Stands for |
| SDD | Software design document |
| CD | Class diagram |
| SD | Sequence diagram |
| UI | User Interface |
| WRMS | WIL Report Management System |
| SRS | Software Requirement Specification |

**Definition**

|  |  |
| --- | --- |
| Software Design Document | A description of software created to facilitate analysis, planning, implementation, and decision-making. This design description is -used as a medium for communicating software design information and can be considered as a blueprint or model of the system. |
| Sequence diagram | An interaction diagram that shows how objects operate with one another and in what order. It is a construct of a message sequence chart. |
| Class diagram | A type of static structure diagram that describes the structure of a system by showing the system's classes, their attributes, operations (or methods), and the relationships among objects |
| User Interface | The industrial design field of human–computer interaction, is the space where interactions between humans and machines occur |
| Software Requirement Specification | A description of a software system to be developed. It lays out functional and non-functional requirements, and may include a set of use cases that describe user interactions that the software must provide. |

# **Chapter II | Class diagram**

## **System class diagram**

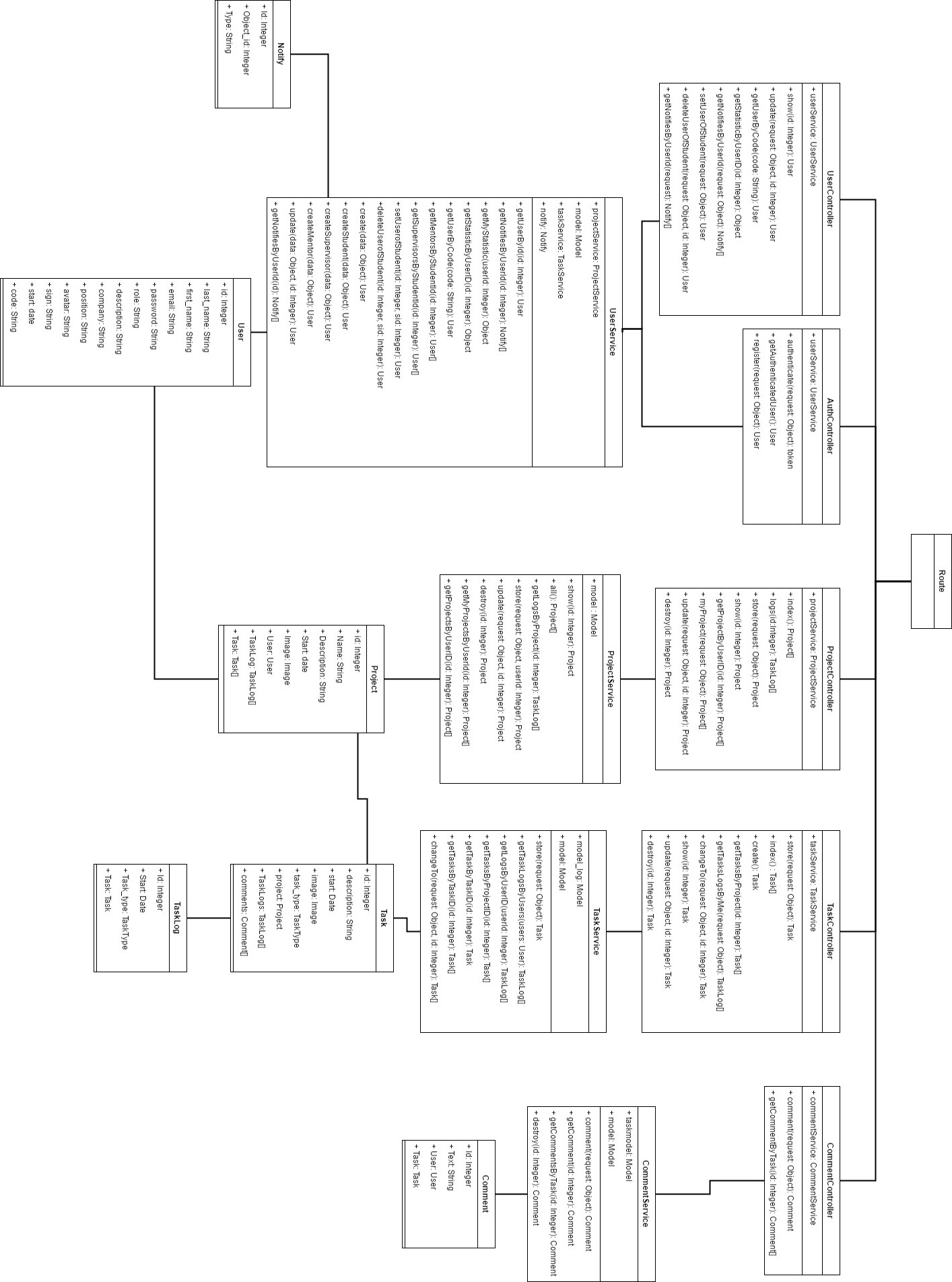
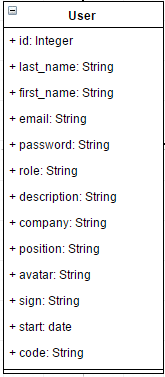


Figure 1 Class diagram of WRMS

## **Class diagram description**

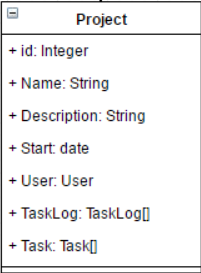
### **Users**



Entities

|  |  |  |  |
| --- | --- | --- | --- |
| ID | Name | Description | Type |
|  | id | Storing an id for identifying the user | int |
|  | first\_name | Storing a first name of the user | String |
|  | last\_name | Storing a last name of the user | String |
|  | email | Storing an email of the user | String |
|  | password | Storing a password of the user | String |
|  | description | Storing a description of the user | String |
|  | company | Storing a company of the user | String |
|  | position | Storing a position of the user | String |
|  | avatar | Storing a path of profile image | String |
|  | sign | Storing a sign of user | String |
|  | start | Storing a start date of user | Date |
|  | role | Storing a role of user | String |
|  | code | Storing a code of the user | String |

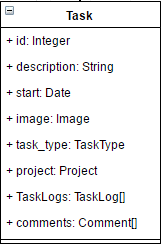
### **Project**



Entities

|  |  |  |  |
| --- | --- | --- | --- |
| ID | Name | Description | Type |
|  | id | Storing an id for identifying the project | int |
|  | Name | Storing a name of the project | String |
|  | Description | Storing description of the project | String |
|  | Start | Storing the start date of the project | Date |
|  | User | Relate the user object with project object by user id | User |
|  | TaskLog | Storing a list of TaskLog | TaskLog[] |
|  | Task | Storing a list of tasks | Task[] |

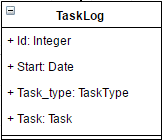
### **Task**



Entities

|  |  |  |  |
| --- | --- | --- | --- |
| ID | Name | Description | Type |
|  | id | Storing an id of the task | int |
|  | description | Storing a description of the task | String |
|  | start | Storing a start date of the task | Date |
|  | image | Storing an image of the task | Image |
|  | task\_type | Storing a type of the task | TaskType |
|  | project | Storing a project of task | Project |
|  | TaskLogs | Storing the history of the task | TaskLog[] |
|  | comment | Storing comments of task | Comment[] |

### **TaskLog**



Entities

|  |  |  |  |
| --- | --- | --- | --- |
| ID | Name | Description | Type |
|  | Id | Storing an id of the TaskLog object | int |
|  | Start | Storing the start date of the TaskLog | Date |
|  | Task\_type | Storing the type of the TaskLog | TaskType[] |
|  | Task | Storing the task of TaskLog | Task |

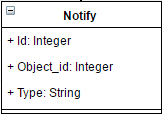
### **Comment**



Entities

|  |  |  |  |
| --- | --- | --- | --- |
| ID | Name | Description | Type |
|  | Id | Storing an id of the comment | int |
|  | Text | Storing a text of the comment | String |
|  | User | Storing the user of the comment | User |
|  | Task | Storing a task of the comment | Task |

### **Notify**



Entities

|  |  |  |  |
| --- | --- | --- | --- |
| ID | Name | Description | Type |
|  | Id | Storing an id of the Notify object | int |
|  | Object\_id | Storing an object id of Notify object | int |
|  | Type | Storing a type of Notify object | String |

### **UserService**



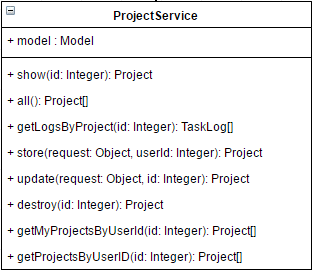
Entities

|  |  |  |  |
| --- | --- | --- | --- |
| ID | Name | Description | Type |
|  | projectService | An object for managing a project for example create, delete project | projectService |
|  | model | An object that contains the user data, provides access to that data, and implements logic to manipulate the data. | model |
|  | taskService | An object for managing a task for example create, delete project | taskService |
|  | notify | An object for storing notification of a user | notify |

Method Detail

|  |  |  |  |
| --- | --- | --- | --- |
| Method name | Description | Parameters | Return |
| getUserById(id: Int): User | This method uses for getting the user by id. | id: int | User |
| If id is null or is not existed in the database will return null. | null |
| getNotifiesByUserID(id Int):Notify[] | This method uses for getting the notifications of specific user by id | id: int | Notify[] |
| If id is null or is not existed in the database will return null. | null |
| getMyStatistic(userId):Object | This method uses for getting their own statistics (all projects, all tasks) of the user | userId: int | {  total\_projects Int,  total\_tasks: Int  } |
| If id is not existed in the database | {  total\_projects = 0,  total\_tasks = 0  } |
| getStatisticByUserID(id: Int): Object | This method uses for getting the statistics (all projects, all tasks and days) of specific user by id | Id: int | {  total\_projects Int,  total\_tasks: Int  } |
| If id is not existed in the database | {  total\_projects = 0,  total\_tasks = 0  } |
| getUserByCode(code: String): User | This method uses for getting the user object by the code of user | code: String | User |
| If code is null or is not existed in the database will return null. | null |
| getMentorsByStudentId(id: Int): User[] | This method uses for getting the list of mentors by the student ID | id: int | User[] |
| If id is null or is not existed in the database will return null. | null |
| getSupervisorsByStudentId(id: Int): User[] | This method uses for getting the list of supervisors by the student ID | id: int | User[] |
| If id is null or is not existed in the database will return null. | null |
| setUserOfStudent(id: Int, sid: Int): User | This method uses for matching mentor and supervisor with student by inputting user ID(mentor and supervisor) and Student ID | id: int  sid: int | User |
| If is or sid is null or is not existed in the database will return null. | null |
| deleteUserOfStudent(id: Int, sid: Int):User | This method uses for deleting the matching mentor and supervisor from the student object by student id and mentor or supervisor id | id: int  sid: int | User |
| create(data: Object): User | This method uses for creating the user object depend by role. | data: Object | User |
| If any detail of user is empty will return false. | false |
| createStudent(data: Object): User | This method uses for creating the student object | data: Object | User |
| If any detail of user is empty will return false. | false |
| createSupervisor(data: Object):User | This method uses for creating the supervisor object | data: Object | User |
| If any detail of user is empty will return false. | false |
| createMentor(data: Object):User | This method uses for creating the mentor object | data: Object | User |
| If any detail of user is empty will return false. | false |
| update(data: Object, id: Int): User | This method uses for updating the user object. | data: Object  id: Int | User |
| If any detail of user is empty will return false. | false |
| getNotifiesByUserId(id:Int): Notify[] | This method uses for getting all notification (comment and task status updated) of specific user by user id | id:int | Notify[] |

### **ProjectService**



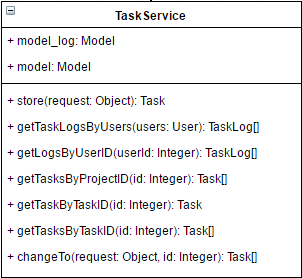
Entities

|  |  |  |  |
| --- | --- | --- | --- |
| ID | Name | Description | Type |
|  | model | An object that contains the project data, provides access to that data, and implements logic to manipulate the data. | Project |

Method Detail

|  |  |  |  |
| --- | --- | --- | --- |
| Method name | Description | Parameters | Return |
| show(id):project | This method uses for getting a project by using project id | Id: Int | Project |
| If id is null or is not existed in the database will return null. | null |
| all():project[] | This method uses for getting all projects | - | Project[] |
| getLogsByProject(id:Int):TaskLog[] | This method uses for getting all tasklogs (activities of task) by the id of project | id:Int | TaskLog[] |
| If id is null or is not existed in the database will return null. | null |
| store(request: Object, userId: Int): Project | This method uses for saving the input data into the project object | request: Object  userId: Int | Project |
| If any detail of project is empty will return false. | false |
| update(request: Object, id: Int): Project | This method uses for updating the input data into the project object from project id | request: Object  id: Int | Project |
| If any detail of project is empty will return false. | false |
| destroy(id:Int):Project | This method uses for deleting the project object by project id | id:Int | Project |
| If id is empty or is not existed in the database will return null. | null |
| getMyProjectsByUserID(id: Int): project[] | This method uses for getting their own project objects search by user id in project. | id: Int | Project[] |
| If id is empty or is not existed in the database will return null | null |
| getProjectByUserId(id: Int): project[] | This method uses for getting all project objects by id of the user in project. | id: Int | Project[] |
| If id is empty or is not existed in the database will return null | null |

### **TaskService**



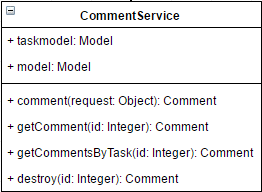
Entities

|  |  |  |  |
| --- | --- | --- | --- |
| ID | Name | Description | Type |
|  | model\_log | An object that contains the task log data, provides access to that data, and implements logic to manipulate the data. | TaskLog |
|  | model | An object that contains the task data, provides access to that data, and implements logic to manipulate the data. | Task |

Method Detail

|  |  |  |  |
| --- | --- | --- | --- |
| Method name | Description | Parameters | Return |
| store(request: Object): Task | This method uses for saving the input data into the task object with ToDo status default | request: Object | Task |
| If any detail of task is empty will return false. | false |
| getTaskLogsByUsers(users: User): TaskLog[] | This method uses for getting all TaskLog (activities of tasks) of all tasks related users by inputting list of user id | users: User | TaskLog[] |
| getLogsByUserID(userId: Int): TaskLog[] | This method uses for getting all TaskLog (activities of tasks) of all tasks related with the specific user by inputting the user id | userId: Int | TaskLog[] |
| getTasksByProjectID(id: Int): Task[] | This method uses for getting all Task related with project by project id | id: Int | Task[] |
| getTaskByTaskID(id: Int): Task | This method uses for getting the task object detail by inputting task ID | id: Int | Task |
| If id is null or is not existed in the database will return null. | null |
| getTaskLogsByTaskID(id: Int): Task[] | This method uses for getting all TaskLog related with task by inputting task ID | id: Int | Task[] |
| changeTo(request: Object, id: Int): Task | This method uses for changing the status of specific task by task id and create new task log with changed status | request: Object  id: Int | Task |

### **CommentService**



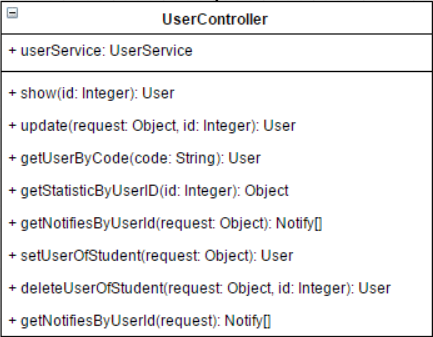
Entities

|  |  |  |  |
| --- | --- | --- | --- |
| ID | Name | Description | Type |
|  | taskmodel | An object that contains the task data, provides access to that data, and implements logic to manipulate the data. | Model |
|  | model | An object that contains the comment data, provides access to that data, and implements logic to manipulate the data. | Model |

Method Detail

|  |  |  |  |
| --- | --- | --- | --- |
| Method name | Description | Parameters | Return |
| comment(request: Object): Comment | This method uses for adding a comment to task by comment detail and task id in request | request: Object | Comment |
| If text is empty will return false | false |
| getComment(id: Int): Comment | This method uses for getting a comment detail by comment id | id: Int | Comment |
| If id is null or is not existed in the database will return null. | null |
| getCommentsByTask(id: Int): Comment | This method uses for getting all comment of a specific task by task id | id: Int | Comment |
| If id is null or is not existed in the database will return null. | null |
| destroy(id: Int): Comment | This method uses for deleting the comment from the database by comment id | id: Int | Comment |

### **UserController**



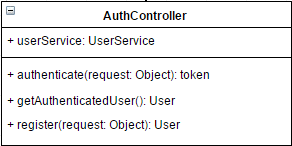
Entities

|  |  |  |  |
| --- | --- | --- | --- |
| ID | Name | Description | Type |
|  | userService | Object of userService, it uses to manage the user object by calling methods from userService class. | UserService |

Method Detail

|  |  |  |  |
| --- | --- | --- | --- |
| Method name | Description | Parameters | Return |
| show(id: Int): User | This method uses for displaying the user object by user id | id: Int | User |
| update(request: Object): User | This method uses for updating the new data of the user by user id | request: Object | User |
| getUserByCode(code: String): User | This method uses for getting the user object by inputting the code of supervisor or mentor. | code: String | User |
| getStatisticByUserID(id: Int): Object | This method uses for getting the statistics (all project, all task and days) of a specific user | id: Int | Object |
| getNotifiesByUserId(request: Object): Notify[] | This method uses for getting the notifications (comment and task status update) of a specific user | request: Object | Notify[] |
| setUserOfStudent(request: Object): User | This method uses for setting the mentor and supervisor of the student. | request: Object | User |
| deleteUserOfStudent(request: Object, id: Int): User | This method uses for deleting the matching mentor or supervisor of the student | request: Object  id: Int | User |
| getNotifiesByUserId (request:Object): Notify[] | This method uses for getting all notification (comment and task status update) of specific user | request: Object | Notify[] |

### **AuthController**



Entities

|  |  |  |  |
| --- | --- | --- | --- |
| ID | Name | Description | Type |
|  | userService | Object of userService, it uses to manage the user object by calling methods from userService class. | UserService |

Method Detail

|  |  |  |  |
| --- | --- | --- | --- |
| Method name | Description | Parameters | Return |
| authenticate(request: Object): token | This method uses for receiving username and password from ‘request object’, then find the use from database. And generate token and return with JWT | request: Object | token |
| getAuthenticatedUser(): User | This method uses for getting the authenticated user object by JWT in header request | - | User |
| register(request: Object): User | This method uses for receiving the user information which called ‘request’, then create the user object in the database. | request: Object | User |

### **ProjectController**



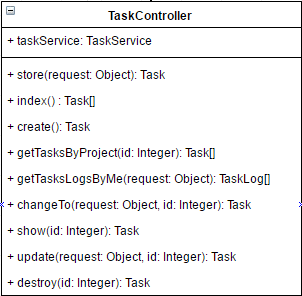
Entities

|  |  |  |  |
| --- | --- | --- | --- |
| ID | Name | Description | Type |
|  | projectService | Object of projectService, it uses to manage the project object by calling methods from projectService class. | ProjectService |

Method Detail

|  |  |  |  |
| --- | --- | --- | --- |
| Method name | Description | Parameters | Return |
| index(): Project[] | This method uses for getting all project object | request: the object which contain the owner data. | Project[] |
| logs(id:Int): TaskLog[] | This method uses for getting logs of specific project by using project id | Id: Int | TaskLog[] |
| store(request: Object): Project | This method uses for storing the project data which called ‘request’ object in the database. | request: Object | Project |
| show(id: Int): Project | This method uses for displaying the project object data which are project name, project description, start date, image, creator name, and tasks. | id: Int | Project |
| getProjectByUserID(id: Int): Project[] | This method uses for getting the list of the projects related user id in project by inputting userID | in: Int | Project[] |
| myProject(id: Int): Project[] | This project uses for getting the list of the projects of the owner by user id in request convert from JWT. | id: Int | Project[] |
| update(request: Object, id: Int): Project | This method uses for updating the project object data by project id | request: Object  id: Int | Project |
| destroy(id: Int): Project | This method uses for removing the project object by project id | id: Int | Project |

### **TaskController**



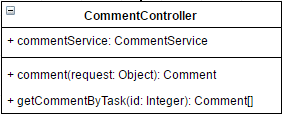
Entities

|  |  |  |  |
| --- | --- | --- | --- |
| ID | Name | Description | Type |
|  | taskService | Object of taskService, it uses to manage the task object by calling methods from taskService class. | TaskService |

Method Detail

|  |  |  |  |
| --- | --- | --- | --- |
| Method name | Description | Parameters | Return |
| store(request: Object): Task | This method uses for creating a task and create new task log with ToDo status default | request: Object | Task |
| index() : Task[] | This method uses for getting all tasks in the database | - | Task[] |
| getTasksByProject(id: Int): Task[] | This method uses for getting all tasks of a specific project by project id | id: Int | Task[] |
| getTasksLogsByMe(request: Object): TaskLog[] | This method uses for getting all tasks logs of the user by user id in request convert in JWT | request: Object | TaskLog[] |
| changeTo(request: Object, id: Int): Task | This method uses for changing the status of specific task and create new task log for save activity by task id | request: Object  id: Int | Task |
| show(id: Int): Task | This method uses for displaying the data of the specific task by task id | id: Int | Task |
| update(request: Object, id: Int): Task | This method uses for updating the data of a task by task detail and task id | request: Object  id: Int | Task |
| destroy(id: Int): Task | This method uses for deleting the task by task id | id: Int | Task |

### **CommentController**



Entities

|  |  |  |  |
| --- | --- | --- | --- |
| ID | Name | Description | Type |
|  | commentService | Object of commentService, it uses to manage the comment object by calling methods from commentService class. | CommentService |

Method Detail

|  |  |  |  |
| --- | --- | --- | --- |
| Method name | Description | Parameters | Return |
| comment(request: Object): Comment | This method uses for adding a comment with task id and user id in request | request: Object | Comment |
| getCommentByTask(id: int): Comment[] | This method uses for getting all comment of a specific task by task id | id: int | comment [] |

## **Sequence diagram**

### **2.3.1 Feature 1**

1. **Register**

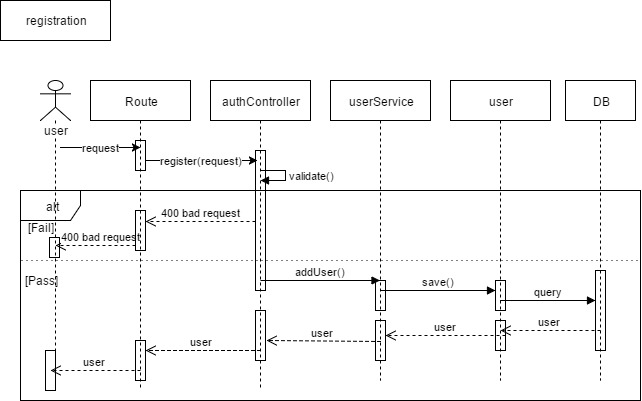


Figure 2 SD-01 Registration

1. **Edit profile**

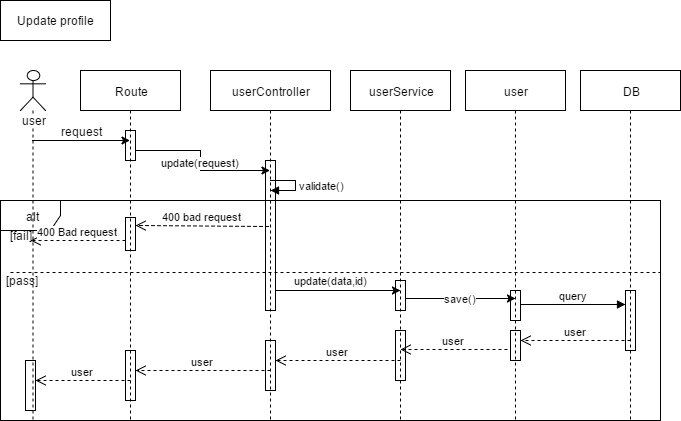


Figure 3 SD-02 Edit profile

1. **Login**

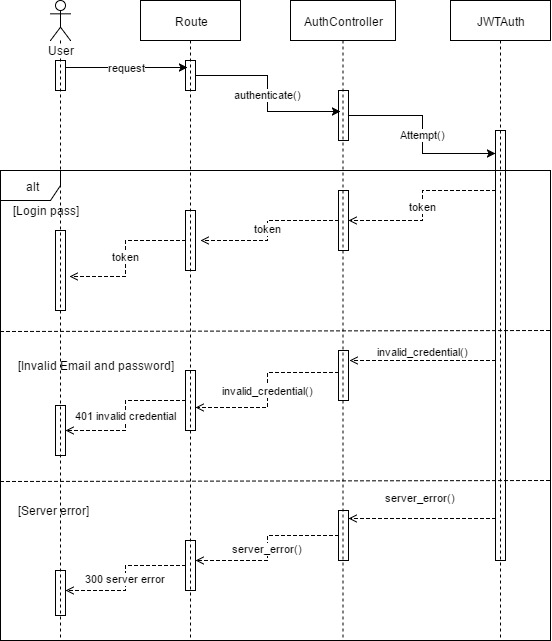


Figure 4 SD-03 Login

1. **Logout**

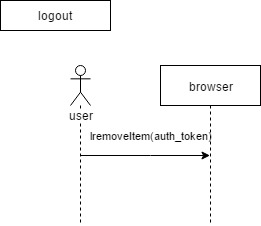


Figure 5 SD-04 Logout

1. **Add a code of supervisor or mentor**

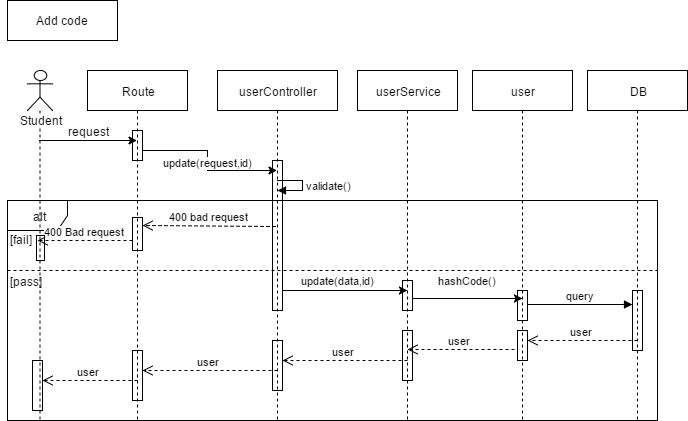


Figure 6 SD-05 Add code

### **2.3.2 Feature 2**

1. **View a task**

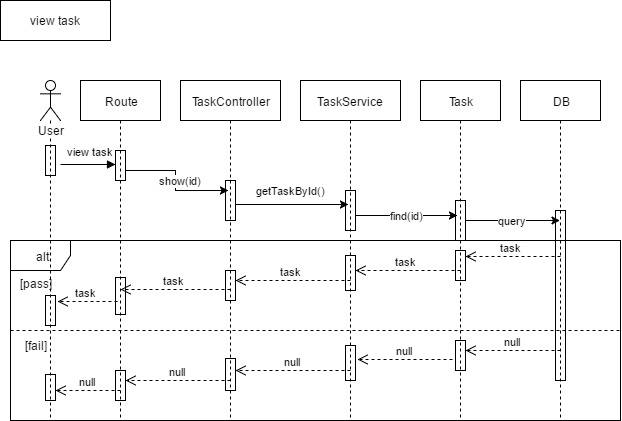


Figure 7 SD-06 View task

1. **View Statistics**

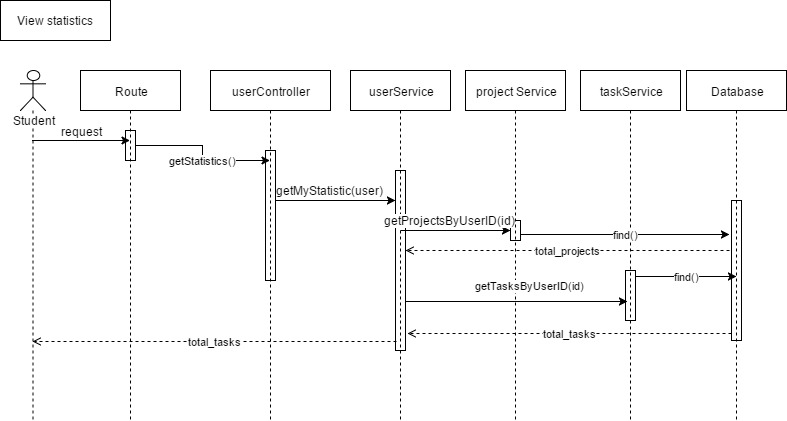


Figure 8 SD-07 View Statistics

1. **Create a project**

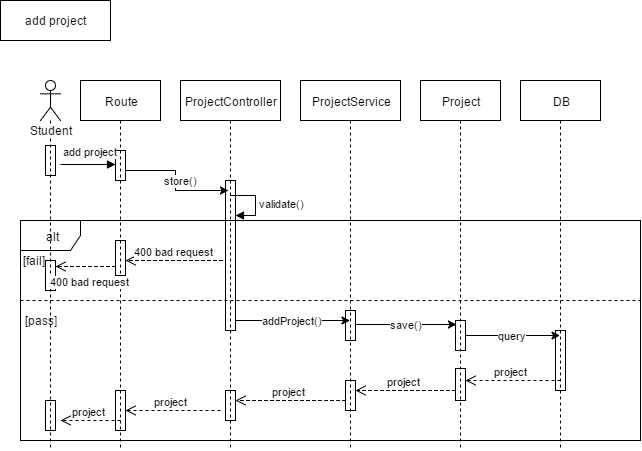


Figure 9 SD-08 Create project

1. **Edit a project**

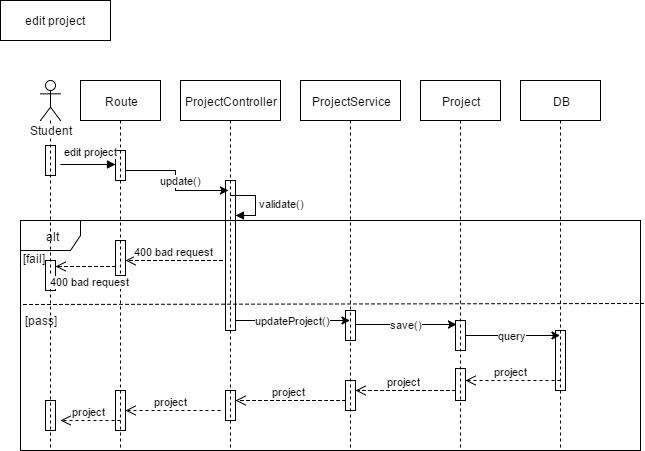


Figure 10 SD-09 Edit project

1. **Delete a project**



Figure 11 SD-10 Delete project

1. **Create a task**

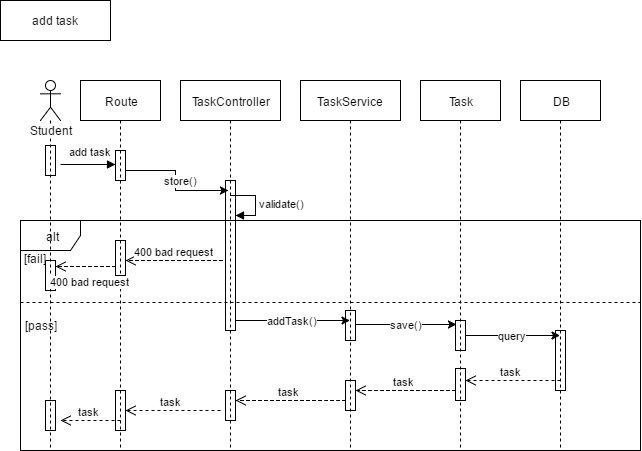


Figure 12 SD-11 Create task

1. **Edit a task**



Figure 13 SD-12 Edit task

1. **Delete a task**

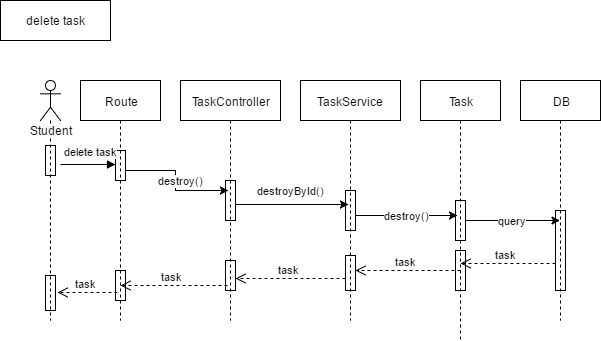


Figure 14 SD-13 Delete task

1. **Change a status of a task**

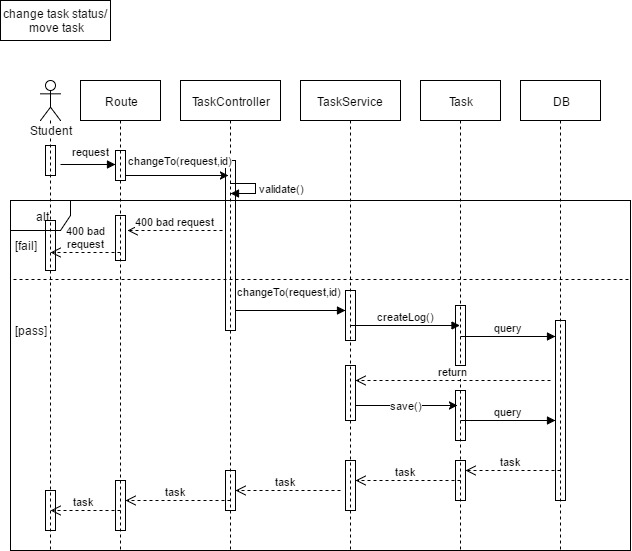


Figure 15 SD-14 Change a status of a task

### 2.3.3 Feature 3

1. **View a comment**

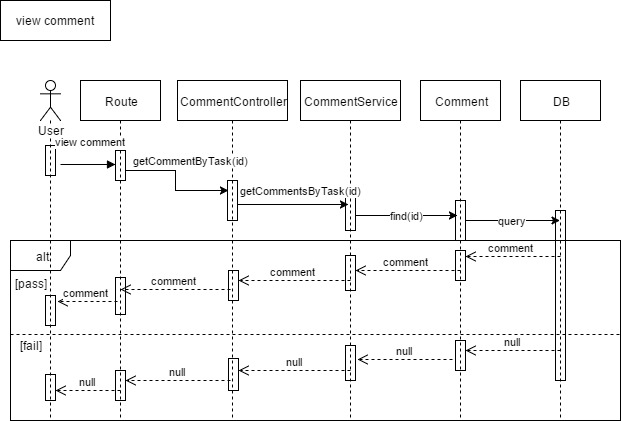


Figure 16 SD-15 View Comment

1. **Create a comment**

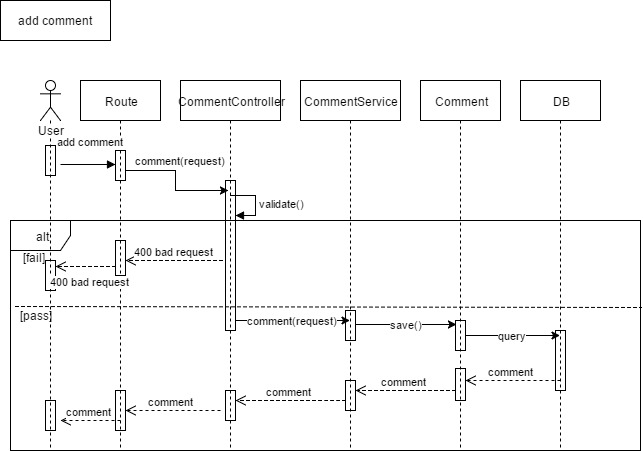


Figure 17 SD-16 Create comment

1. **Delete a comment**

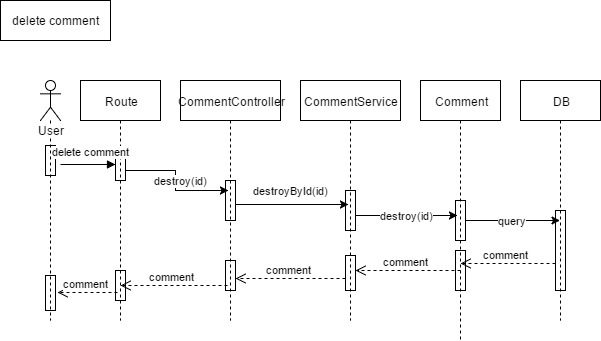


Figure 18 SD-17 Delete comment

1. **View activities of tasks in a specific project**

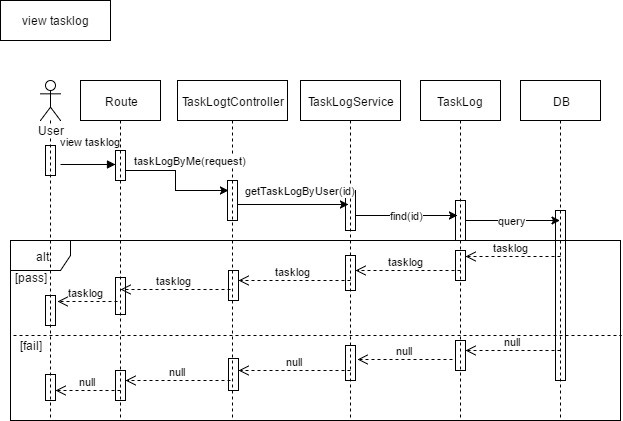


Figure 19 SD-18 View activities in a project

1. **view activities of all under guidance students**

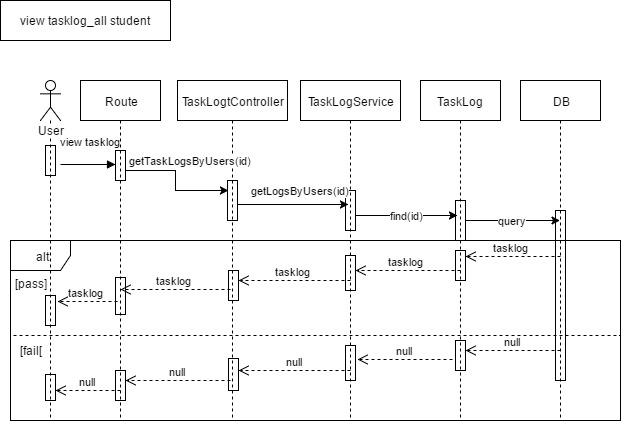


Figure 20 SD-19 View activity of all Students

### 2.3.4 Feature 4

1. **Generate a weekly report**

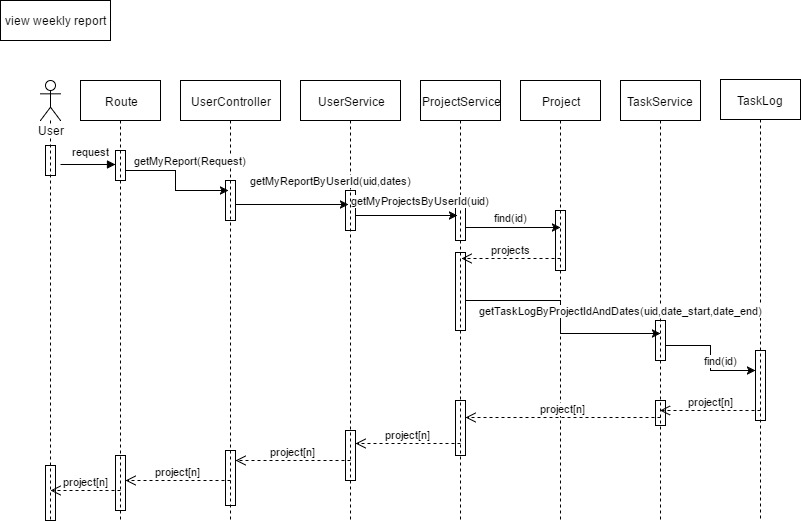


Figure 21 SD-20 Generate a weekly report

1. **Print a weekly report**

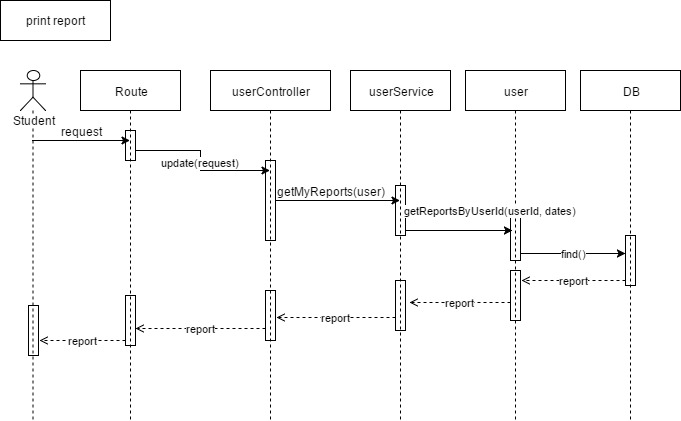


Figure 22 SD-21 Print a weekly report

### 2.3.5 Feature 5

1. **Receive a web notification**

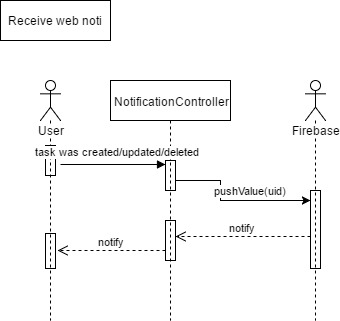


Figure 23 SD-22 Receive web notification

1. **Receive an email notification**

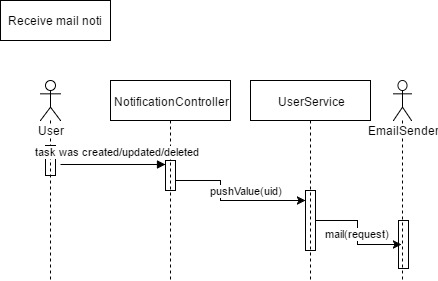


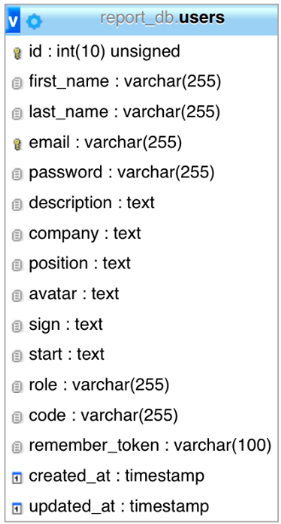
Figure 24 SD-23 Receive Email notification

# **Chapter III | ER diagram**



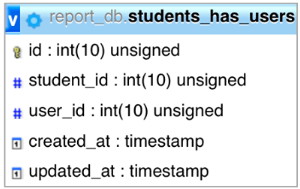
Figure 25 Overview ER-diagram

## **Users table**



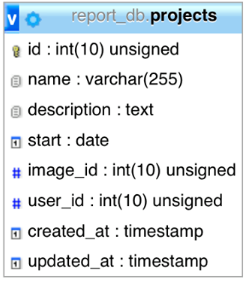
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **No.** | **Field name** | **Type** | **Length** | **PK/FK** |
|  | id | int | 10 | PK |
|  | first\_name | varchar | 255 | - |
|  | last\_name | varchar | 255 | - |
|  | email | varchar | 255 | - |
|  | password | varchar | 255 | - |
|  | description | text | - | - |
|  | company | text | - | - |
|  | position | text | - | - |
|  | avatar | text | - | - |
|  | sign | text | - | - |
|  | start | text | - | - |
|  | role | varchar | 255 | - |
|  | code | varchar | 255 | - |
|  | remember\_token | varchar | 100 | - |
|  | created\_at | timestamp | - | - |
|  | updated\_at | timestamp | - | - |

## **student\_has\_users table**



|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **No.** | **Field name** | **Type** | **Length** | **PK/FK** |
| 1 | Id | int | 10 | PK |
| 2 | Student\_id | Int | 10 | FK |
| 3 | User\_Id | Int | 10 | FK |
| 4 | created\_at | timestamp | - | - |
| 5 | updated\_at | timestamp | - | - |

## **projects table**



|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **No.** | **Field name** | **Type** | **Length** | **PK/FK** |
| 1 | Id | Int | 10 | PK |
| 2 | Name | Varchar | 255 | - |
| 3 | Description | Text | - | - |
| 4 | Start | Date | - | - |
| 5 | Image\_id | Int | 10 | FK |
| 6 | User\_id | Int | 10 | FK |
| 7 | Created\_at | timestamp | - | - |
| 8 | Updated\_at | timestamp | - | - |

## **task table**



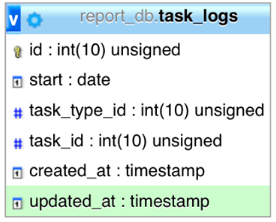
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **No.** | **Field name** | **Type** | **Length** | **PK/FK** |
| 1 | Id | Int | 10 | PK |
| 2 | Name | Varchar | 255 | - |
| 3 | Description | Text | - | - |
| 4 | Start | Date | - | - |
| 5 | Image\_id | Int | 10 | FK |
| 6 | Task\_type\_id | Int | 10 | FK |
| 7 | Project\_id | int | 10 | FK |
| 8 | Created\_at | timestamp | - | - |
| 9 | Updated\_at | timestamp | - | - |

## **task\_types table**



|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **No.** | **Field name** | **Type** | **Length** | **PK/FK** |
| 1 | Id | Int | 10 | PK |
| 2 | Name | varchar | 255 | - |
| 3 | Created\_at | timestamp | - | - |
| 4 | Updated\_at | timestamp | - | - |

## **task\_logs table**



|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **No.** | **Field name** | **Type** | **Length** | **PK/FK** |
| 1 | Id | Int | 10 | PK |
| 2 | Start | Date | - | - |
| 3 | Task\_type\_id | Int | 10 | FK |
| 4 | Task\_id | int | 10 | FK |
| 5 | Created\_at | timestamp | - | - |
| 6 | Updated\_at | timestamp | - | - |

## **comments table**



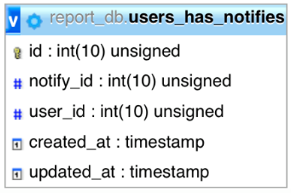
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **No.** | **Field name** | **Type** | **Length** | **PK/FK** |
| 1 | Id | Int | 10 | PK |
| 2 | Text | Text | - | - |
| 3 | User\_id | Int | 10 | FK |
| 4 | Task\_id | int | 10 | FK |
| 5 | Created\_at | timestamp | - | - |
| 6 | Updated\_at | timestamp | - | - |

## **notifies table**



|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **No.** | **Field name** | **Type** | **Length** | **PK/FK** |
| 1 | Id | int | 10 | PK |
| 2 | Obj\_id | Int | 11 | FK |
| 3 | type | varchar | 255 | - |
| 4 | Created\_at | timestamp | - | - |
| 5 | Updated\_at | timestamp | - | - |

## **users\_has\_notifies table**



|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **No.** | **Field name** | **Type** | **Length** | **PK/FK** |
| 1 | Id | int | 10 | PK |
| 2 | Notify\_id | Int | 10 | FK |
| 3 | User\_id | Int | 10 | FK |
| 4 | Created\_at | timestamp | - | - |
| 5 | Updated\_at | timestamp | - | - |

# **Chapter V| UI Design**



Figure 26 UI-01 Login

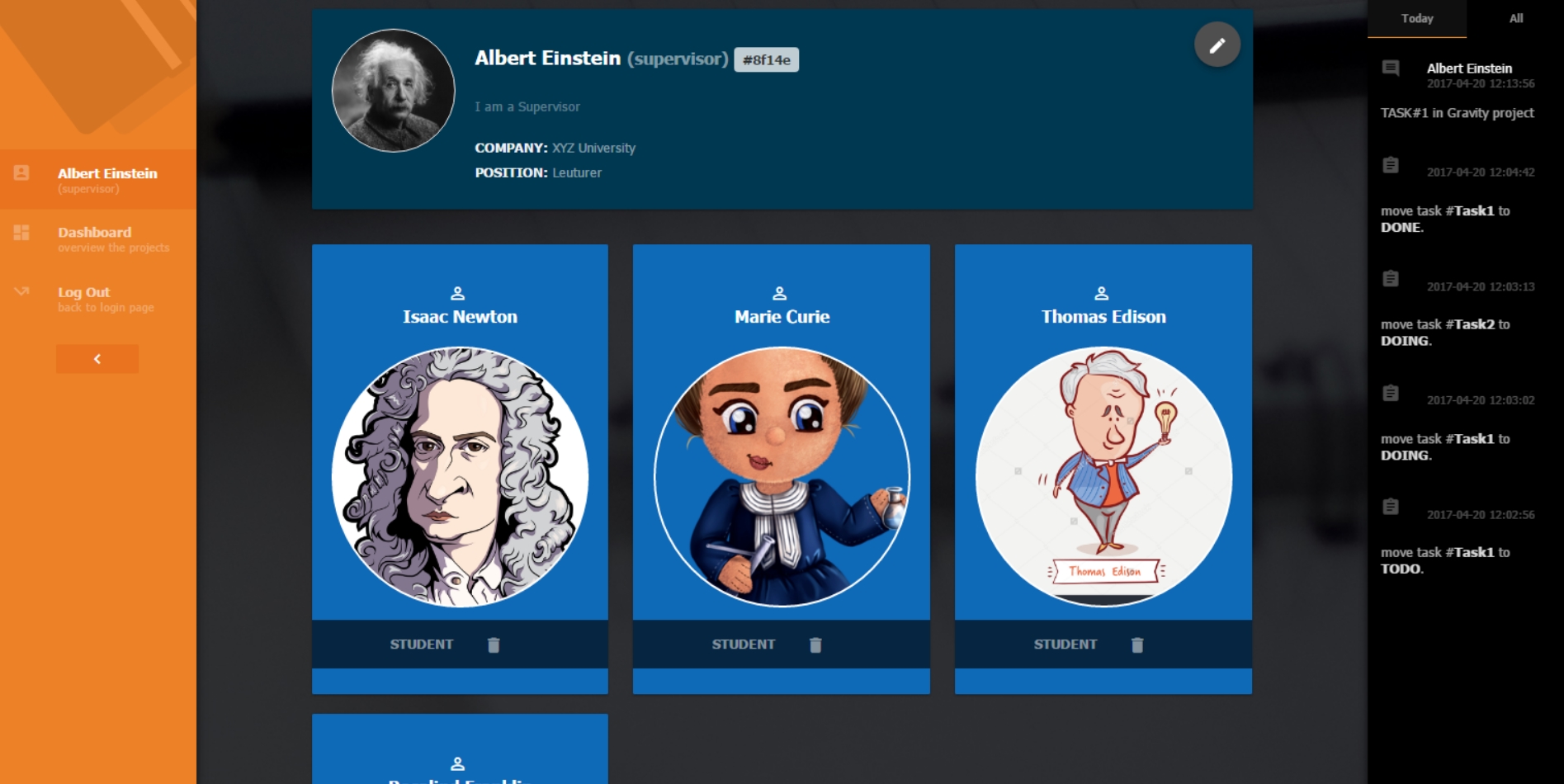


Figure 27 UI-02 Profile

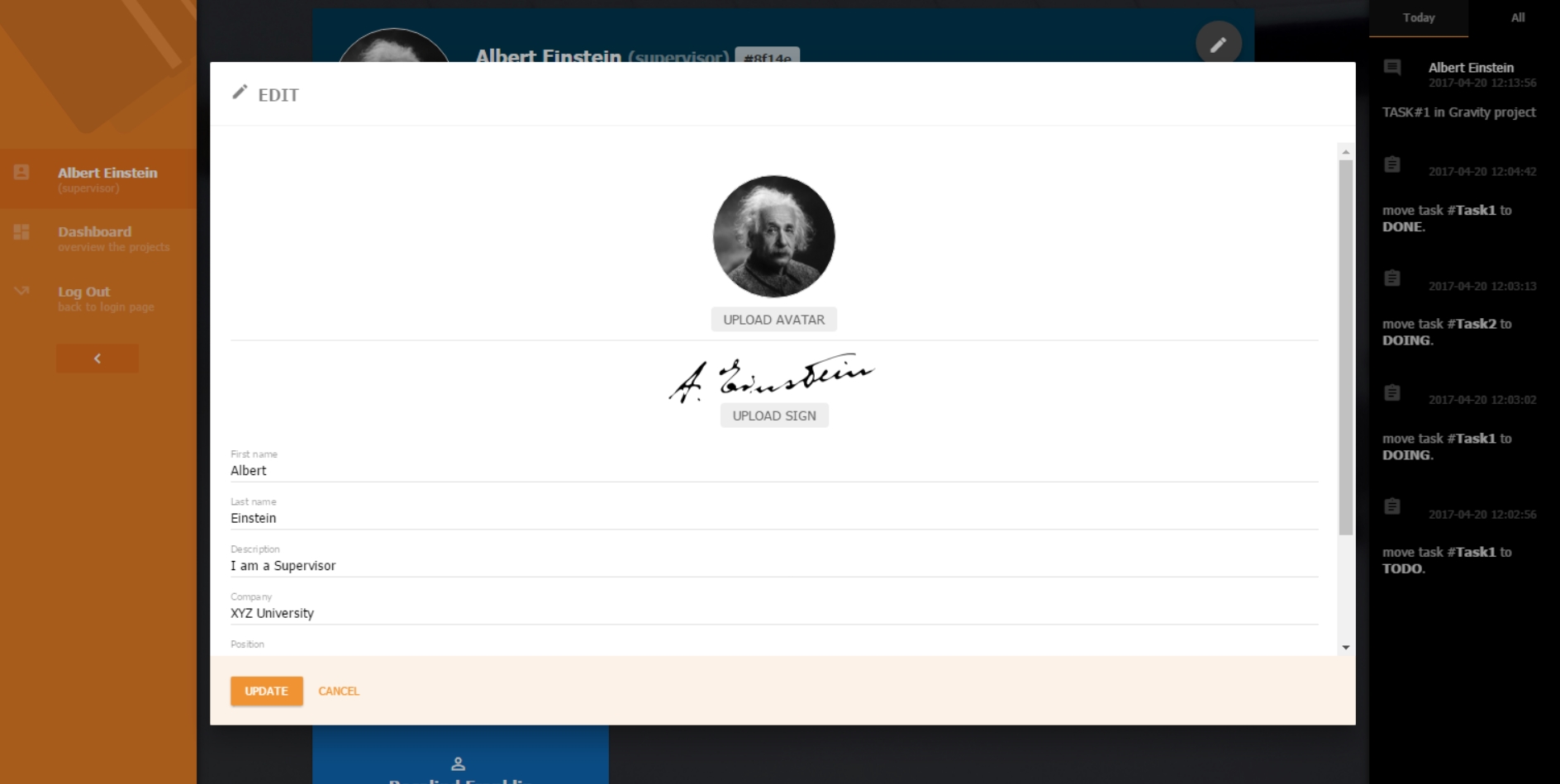


Figure 28 UI-03 Edit profile

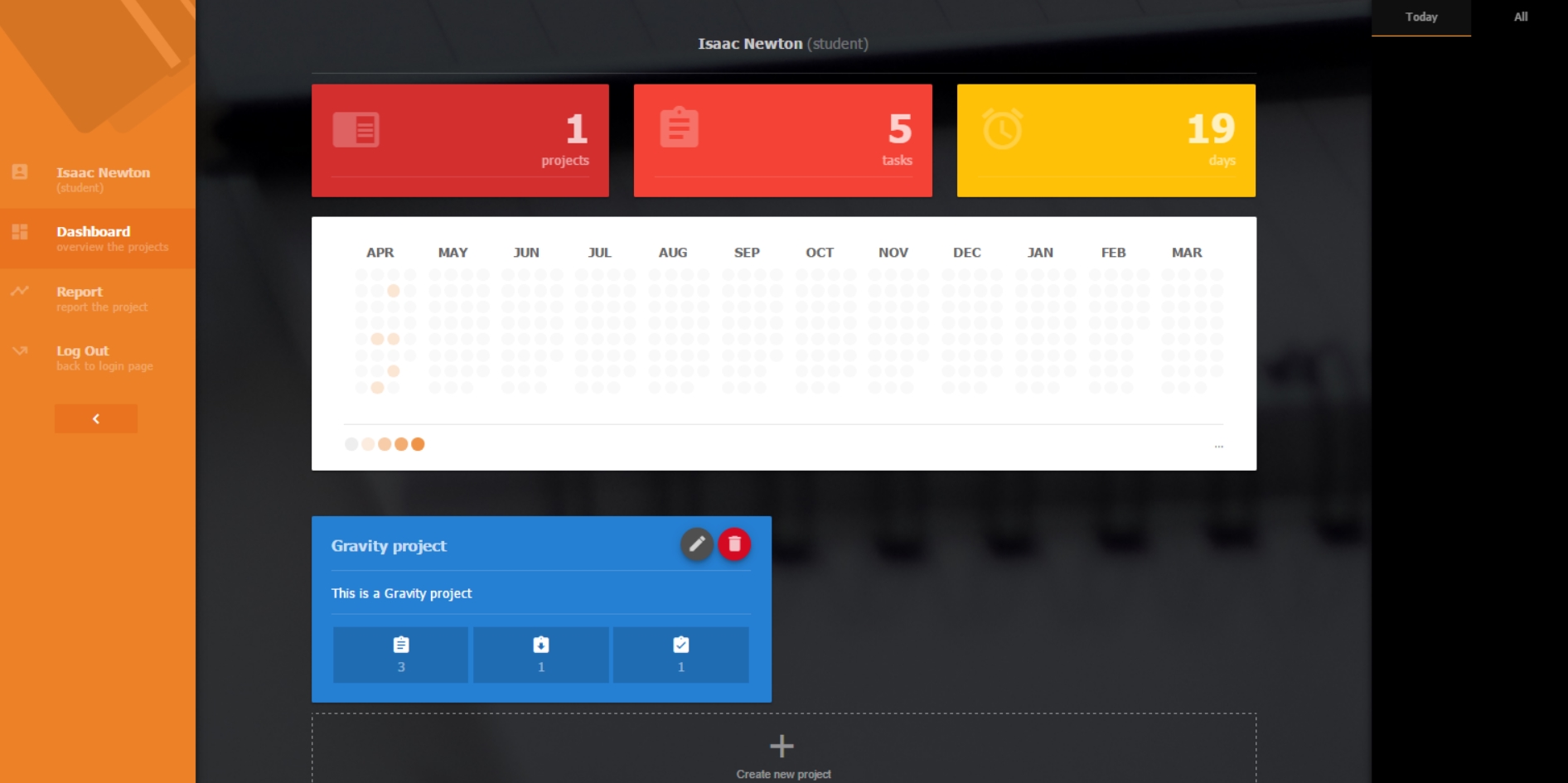


Figure 29 UI-04 Student’s dashboard

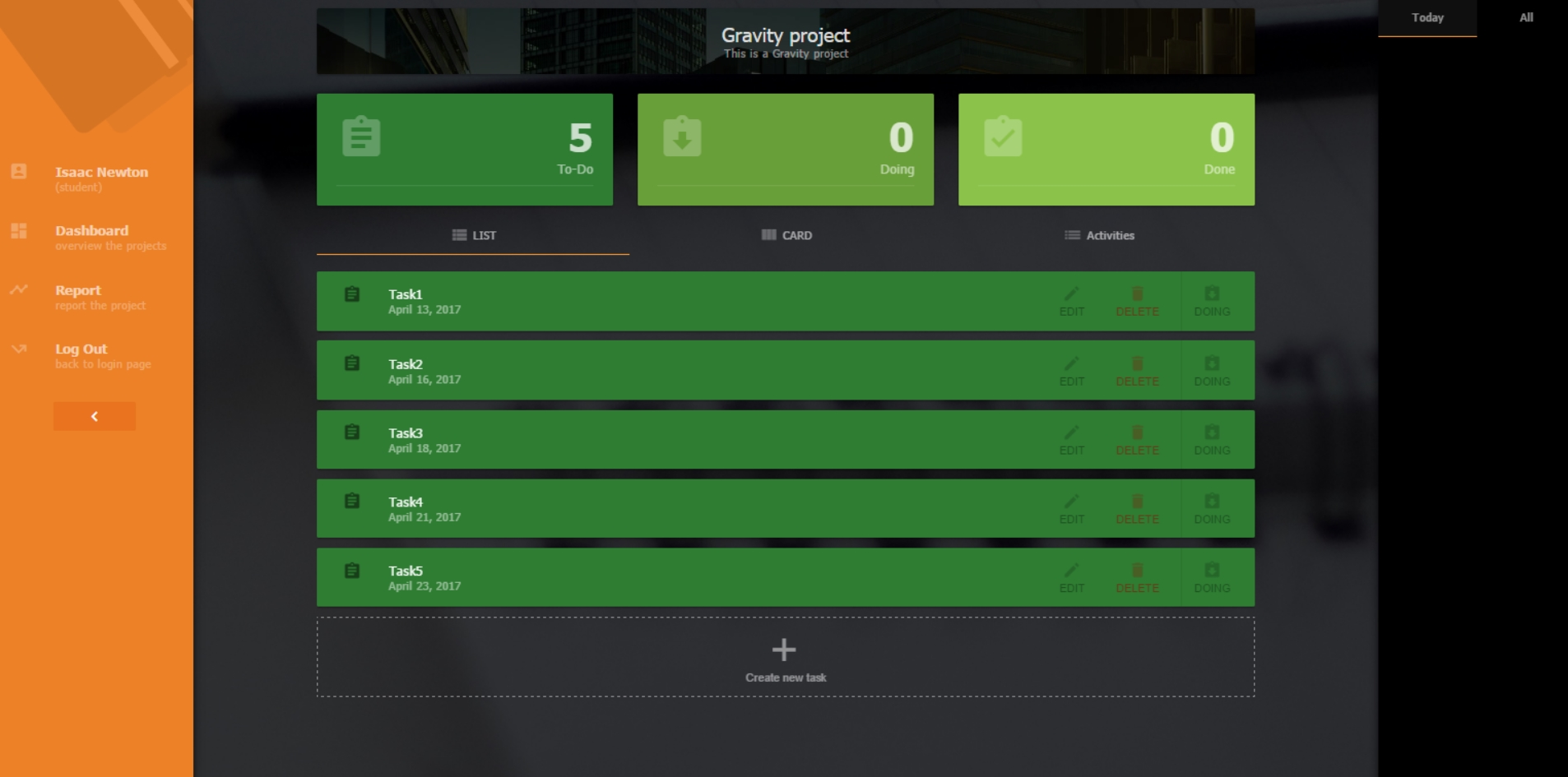


Figure 30 UI-05 Task page (List)



Figure 31 UI-06 Task page (card)

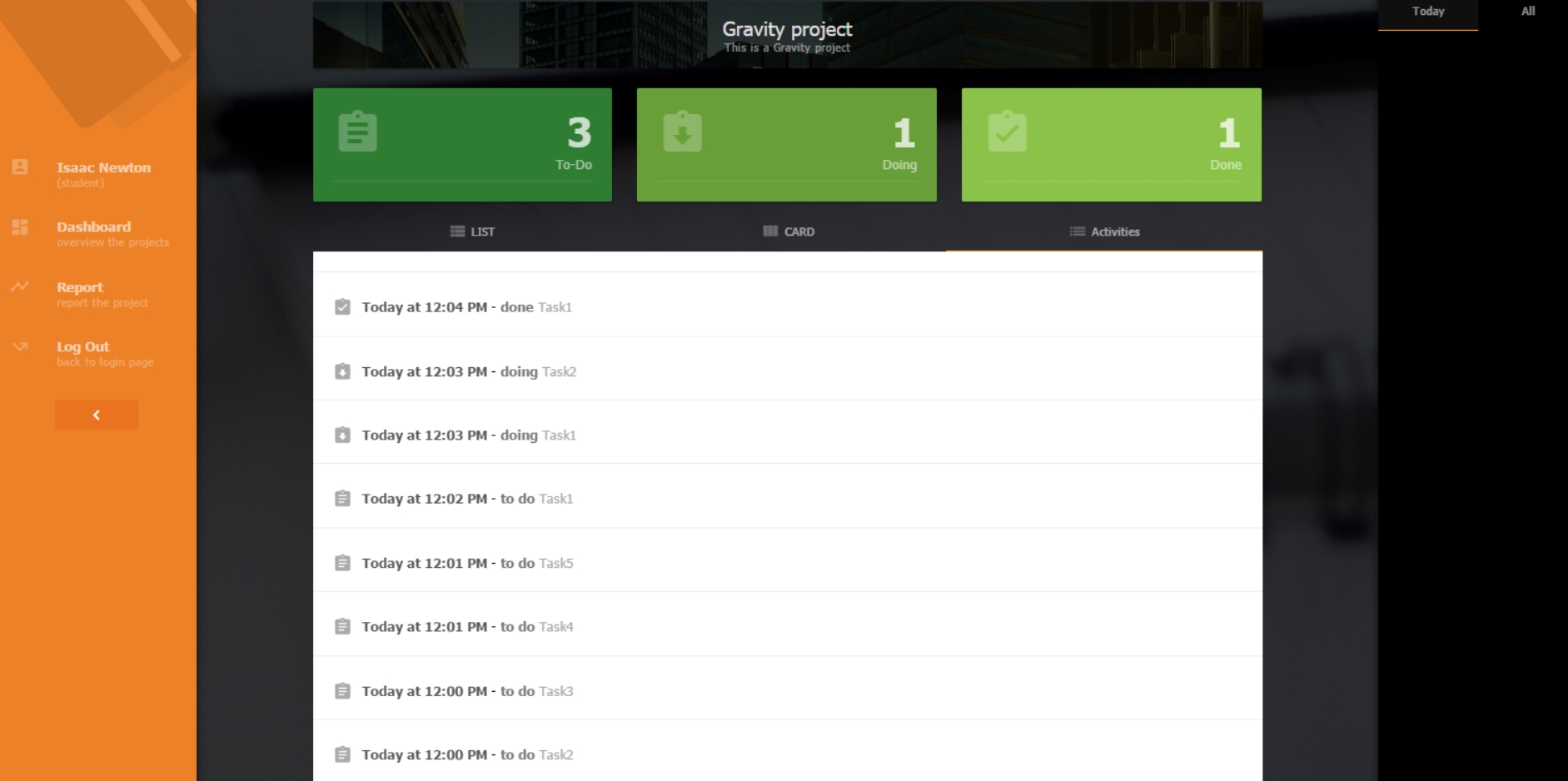


Figure 32 UI-07 Task page (activity)

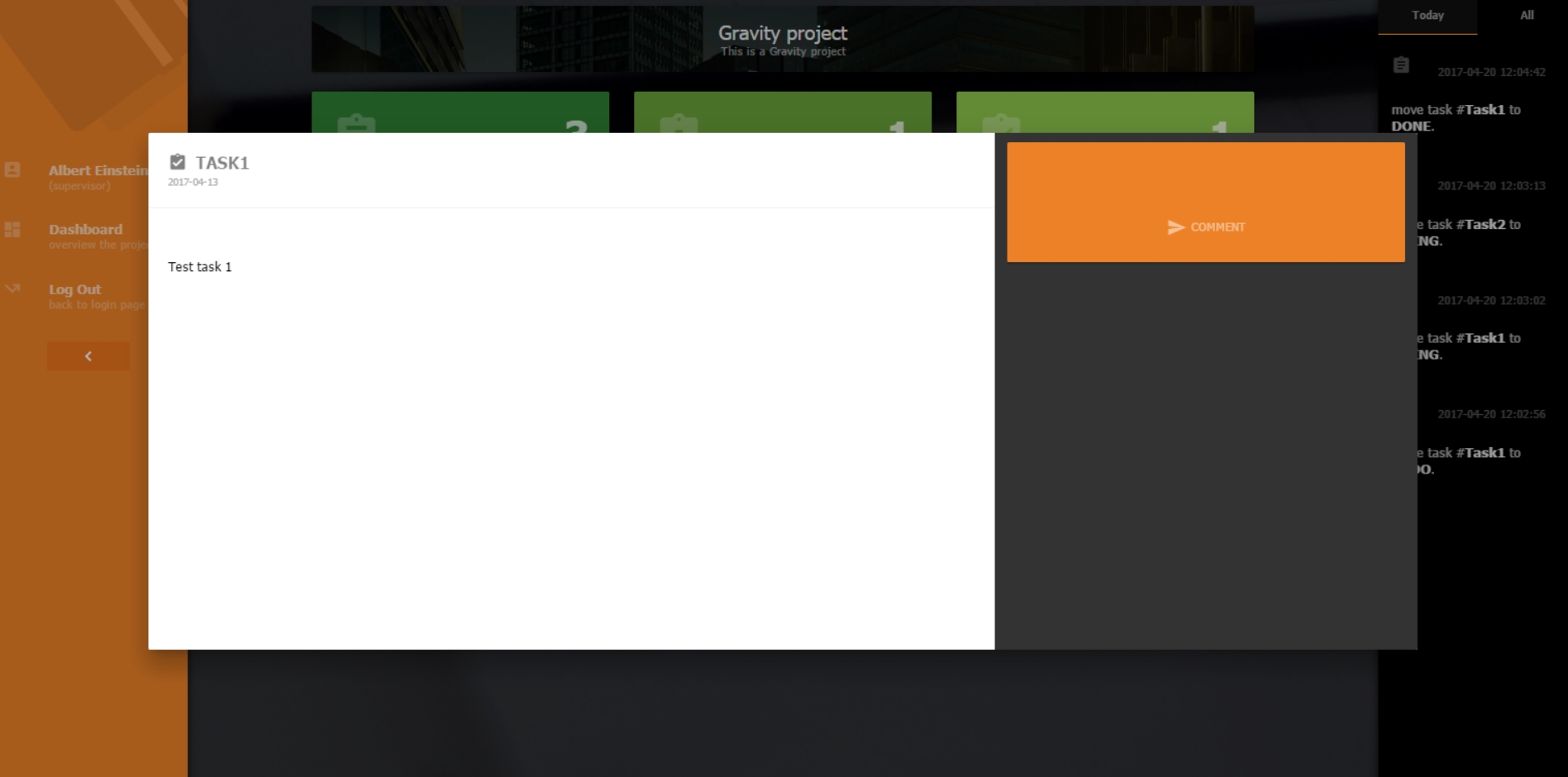


Figure 33 UI-08 Task detail page

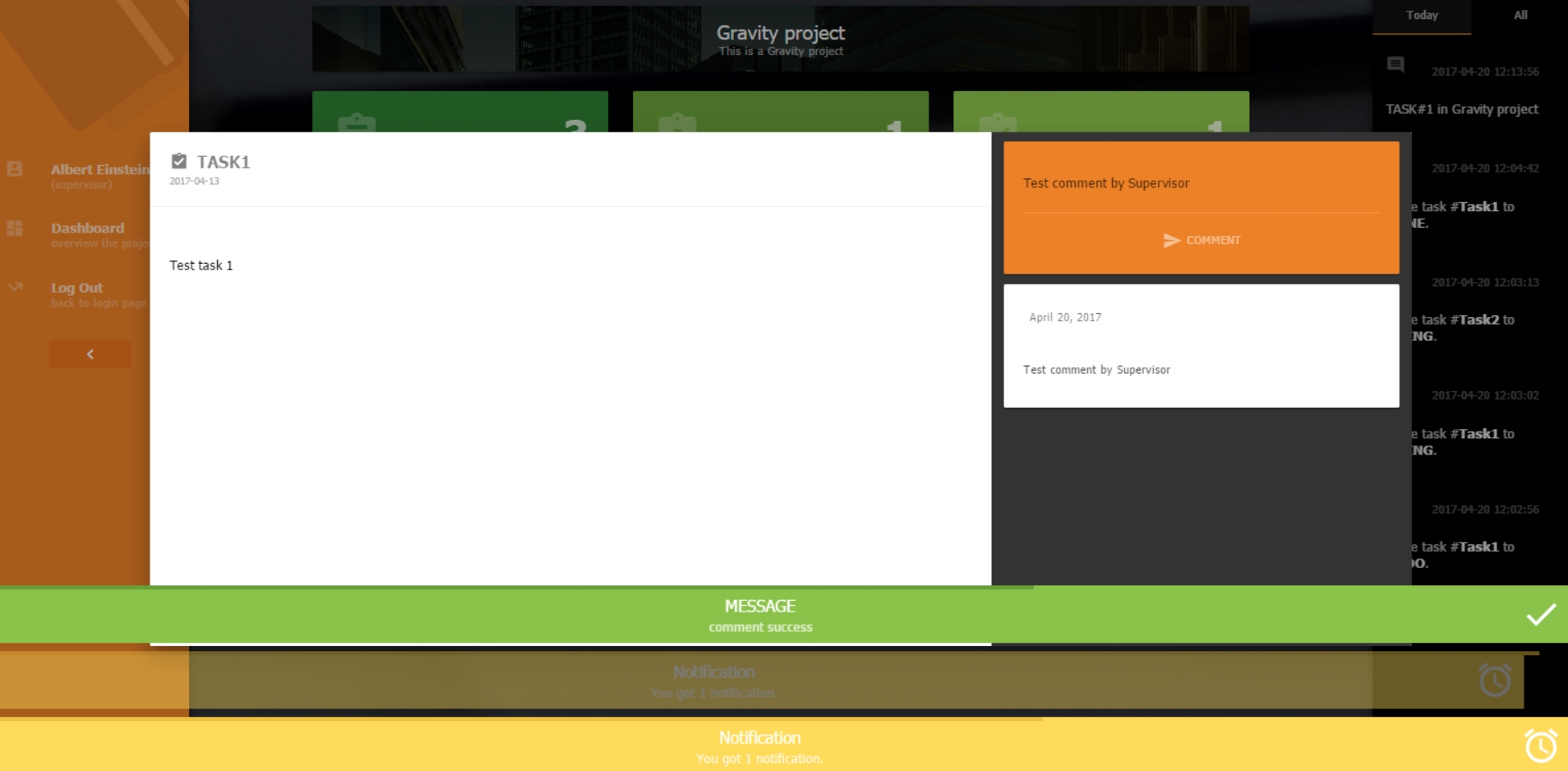


Figure 34 UI-09 Comment and notification

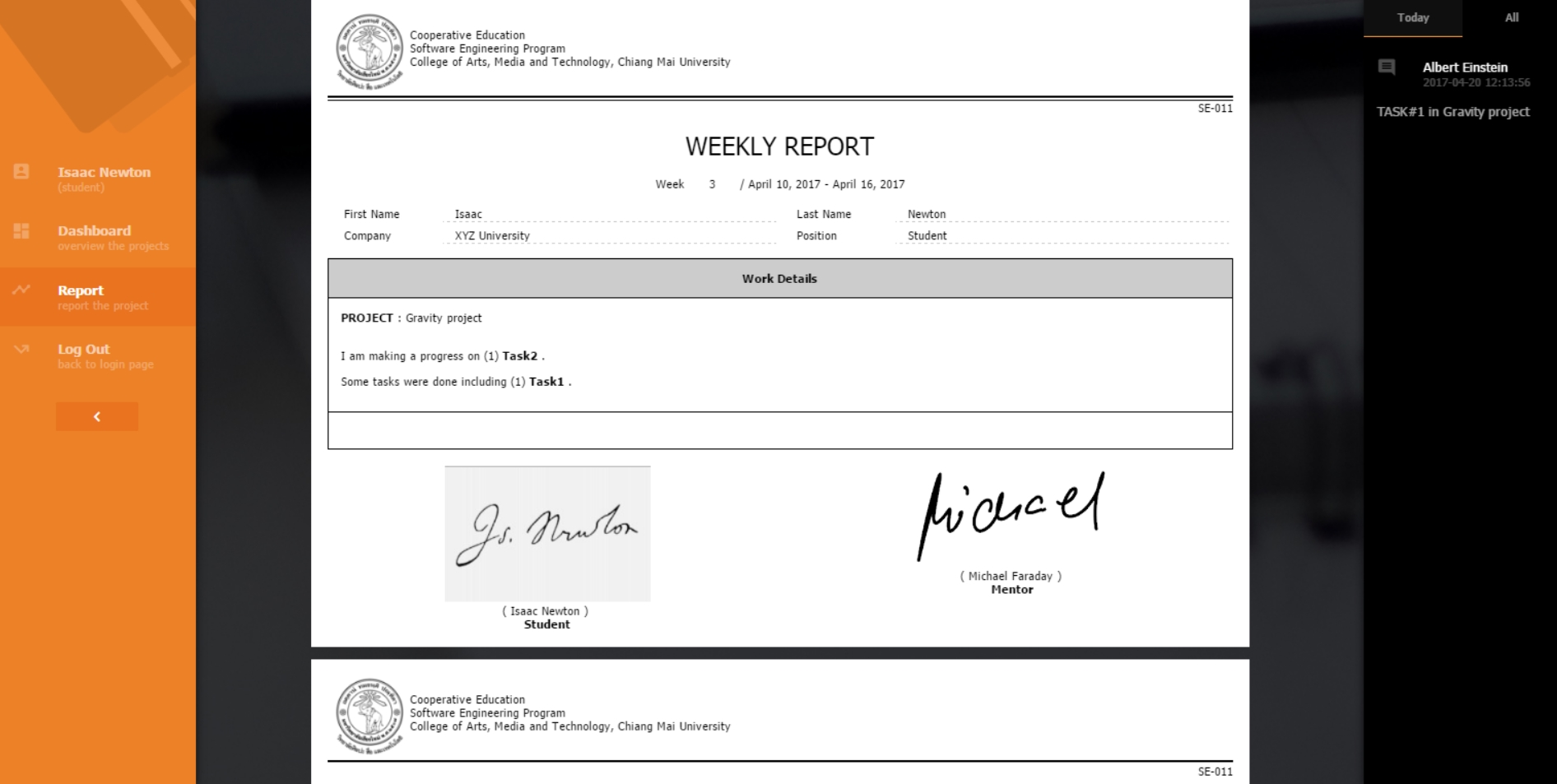


Figure 35 UI-10 report page

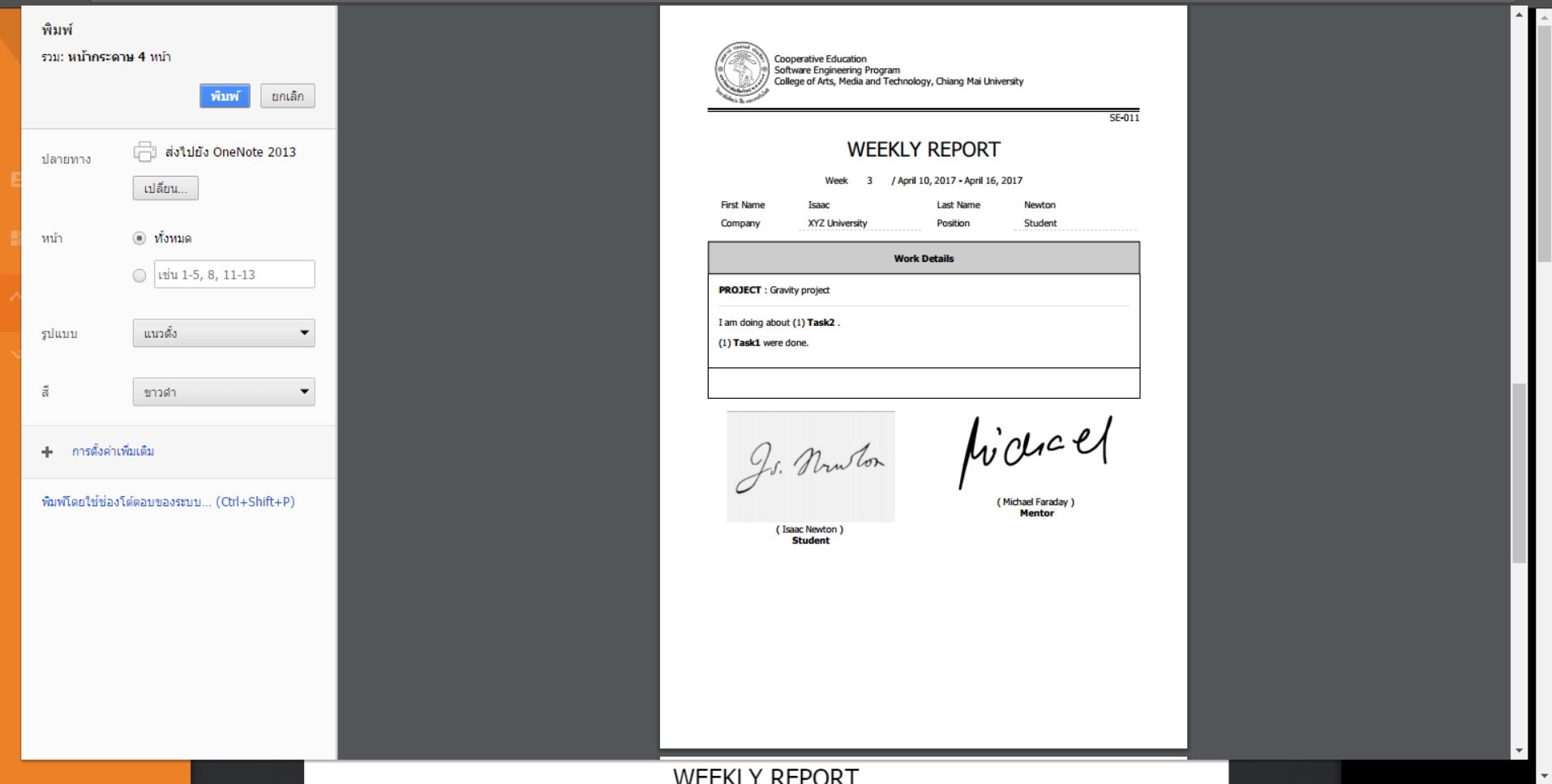


Figure 36 UI-11 Print the weekly report by browser