**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 | PS,VI | PS,VI |
| 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 | PS,VI, | PS,VI |
| WRMS- SDD\_V.0.3.docx | * Edit Chapter III   + Class diagram   + Class diagram description   + Sequence diagram | | Draft | 14-03-2017 | PS,VI | PS,VI, | PS,VI |

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

Table of Contents

[Chapter I | Introduction 6](#_Toc480326524)

[1.1 Purpose 6](#_Toc480326525)

[1.2 Scope 6](#_Toc480326526)

[1.3 Acronyms and Definition 6](#_Toc480326527)

[Chapter II | System Architecture 7](#_Toc480326528)

[Chapter III | Detailed design 8](#_Toc480326529)

[3.1 System class diagram 8](#_Toc480326530)

[3.2 Class diagram description 9](#_Toc480326531)

[CD-01: Users 9](#_Toc480326532)

[CD-02: Project 10](#_Toc480326533)

[CD-03: Task 10](#_Toc480326534)

[CD-04: TaskLog 11](#_Toc480326535)

[CD-05: Comment 11](#_Toc480326536)

[CD-06: Notify 12](#_Toc480326537)

[CD-07: Image 12](#_Toc480326538)

[CD-08: UserService 13](#_Toc480326539)

[CD-09: ProjectService 15](#_Toc480326540)

[CD-10: TaskService 16](#_Toc480326541)

[CD-11: CommentService 17](#_Toc480326542)

[CD-12: NotificationService 18](#_Toc480326543)

[CD-13: UserController 18](#_Toc480326544)

[CD-14: AuthController 19](#_Toc480326545)

[CD-15: ProjectController 20](#_Toc480326546)

[CD-16: TaskController 21](#_Toc480326547)

[CD-17: CommentController 22](#_Toc480326548)

[CD-18: NotificationController 22](#_Toc480326549)

[3.3 Sequence diagram 23](#_Toc480326550)

[3.3.1 Feature 1 23](#_Toc480326551)

[3.3.2 Feature 2 27](#_Toc480326552)

[3.3.3 Feature 3 32](#_Toc480326553)

[3.3.4 Feature 4 34](#_Toc480326554)

[Chapter IV | UI Design 36](#_Toc480326555)

**Table of Figures**

[Figure 1 System architecture 6](#_Toc480465003)

[Figure 2 Class diagram of WRMS 7](#_Toc480465004)

[Figure 3 Sequence diagram of Registration 22](#_Toc480465005)

[Figure 4 Sequence diagram of View profile 23](#_Toc480465006)

[Figure 5 Sequence diagram of Edit profile 24](#_Toc480465007)

[Figure 6 Sequence diagram of Login 25](#_Toc480465008)

[Figure 7 Sequence diagram of Logout 26](#_Toc480465009)

[Figure 8 Sequence diagram of Set code 26](#_Toc480465010)

[Figure 9 Sequence diagram of View project 27](#_Toc480465011)

[Figure 10 Sequence diagram of Add project 27](#_Toc480465012)

[Figure 11 Sequence diagram of Edit project 28](#_Toc480465013)

[Figure 12 Sequence diagram of Delete project 28](#_Toc480465014)

[Figure 13 Sequence diagram of View task 29](#_Toc480465015)

[Figure 14 Sequence diagram of Add task 29](#_Toc480465016)

[Figure 15 Sequence diagram of Edit task 30](#_Toc480465017)

[Figure 16 Sequence diagram of Delete task 30](#_Toc480465018)

[Figure 17 Sequence diagram of Move a task 31](#_Toc480465019)

[Figure 18 Sequence diagram of View TaskLog 31](#_Toc480465020)

[Figure 19 Sequence diagram of View Comment 32](#_Toc480465021)

[Figure 20 Sequence diagram of Add comment 32](#_Toc480465022)

[Figure 21 Sequence diagram of Edit comment 33](#_Toc480465023)

[Figure 22 Sequence diagram of Delete comment 33](#_Toc480465024)

[Figure 23 Sequence diagram of View weekly report 34](#_Toc480465025)

[Figure 24 Sequence diagram of Receive web notification 34](#_Toc480465026)

[Figure 25 Sequence diagram of Receive Email notification 35](#_Toc480465027)

[Figure 26 Login design 36](#_Toc480465028)

[Figure 27 Profile design 36](#_Toc480465029)

[Figure 28 Edit profile design 37](#_Toc480465030)

[Figure 29 Student dashboard design 37](#_Toc480465031)

[Figure 30 Task overview list design 38](#_Toc480465032)

[Figure 31 Task overview card design 38](#_Toc480465033)

[Figure 32 Task activity overview design 39](#_Toc480465034)

[Figure 33 Task detail design 39](#_Toc480465035)

[Figure 34 Comment and notification design 40](#_Toc480465036)

[Figure 35 Generate weekly report 40](#_Toc480465037)

[Figure 36 Print the weekly report by browser 41](#_Toc480465038)

# **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 usecase
* 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 |

# **Chapter II | System Architecture**

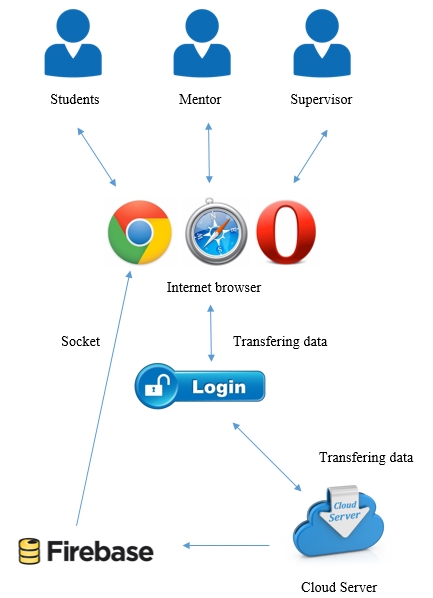


Figure 1 System architecture

# **Chapter III | Detailed design**

## **System class diagram**

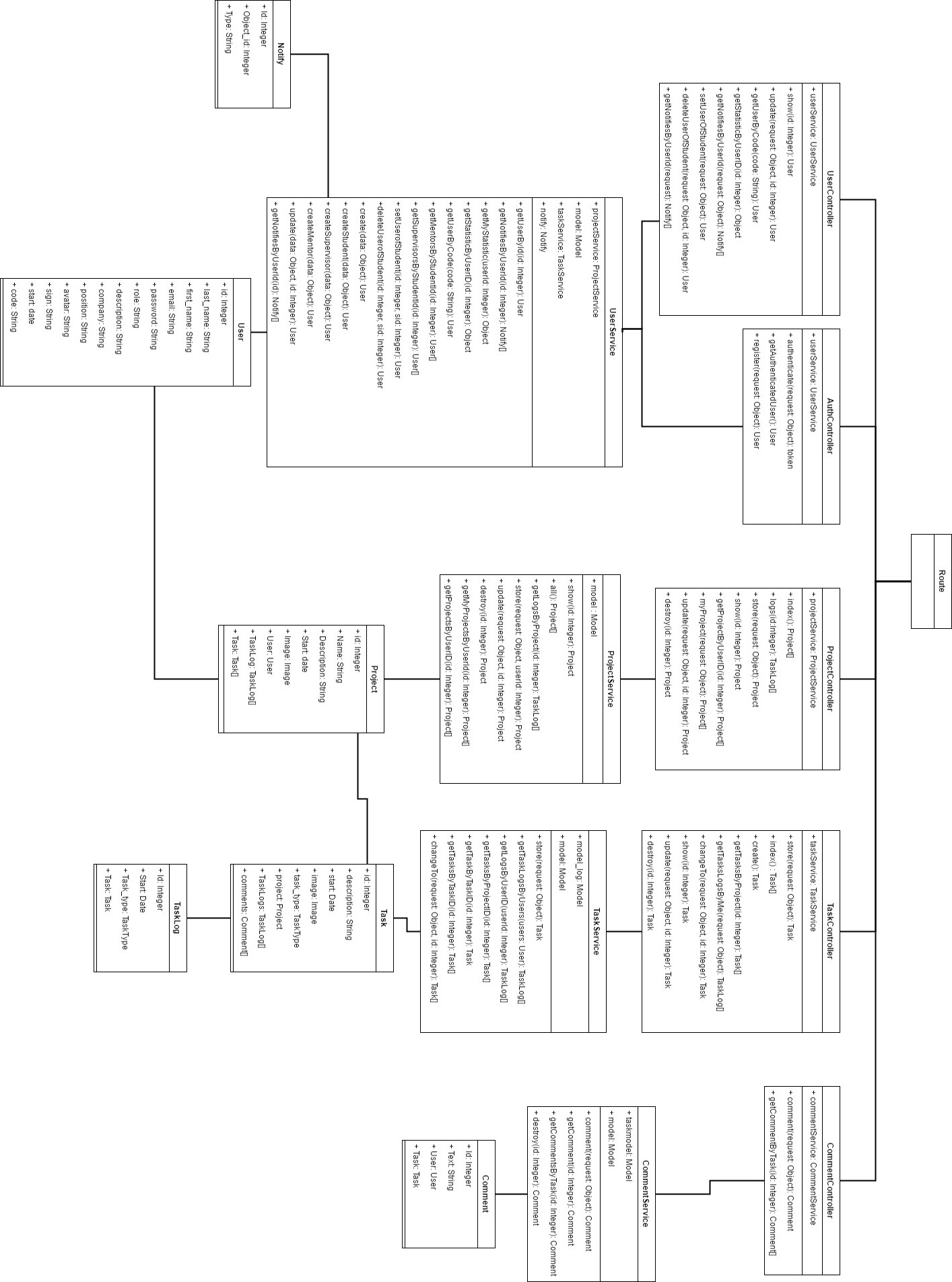
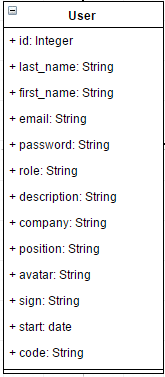


Figure 2 Class diagram of WRMS

## **Class diagram description**

### **Users**



Entities

|  |  |  |  |
| --- | --- | --- | --- |
| ID | Name | Description | Type |
|  | id | Storing an id for identifying the user | Integer |
|  | 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 |

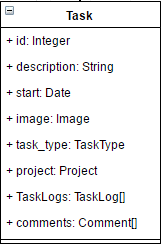
### **Project**



Entities

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

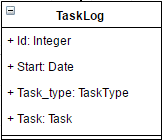
### **Task**



Entities

|  |  |  |  |
| --- | --- | --- | --- |
| ID | Name | Description | Type |
|  | id | Storing an id of the task | Integer |
|  | 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 object |
|  | project | Storing a project of task | Project object |
|  | TaskLogs | Storing the history of the task | Array of TaskLog object |
|  | comment | Storing comments of task | Array of Comment object |

### **TaskLog**



Entities

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

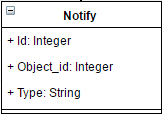
### **Comment**



Entities

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

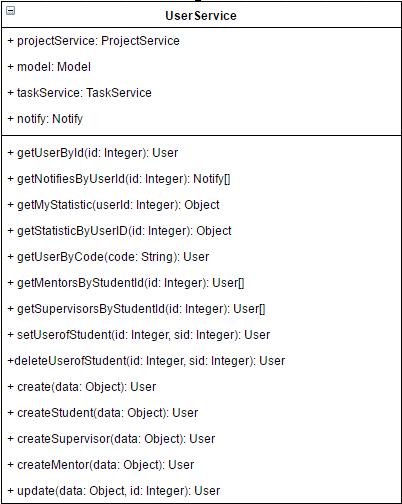
### **Notify**



Entities

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

### **UserService**



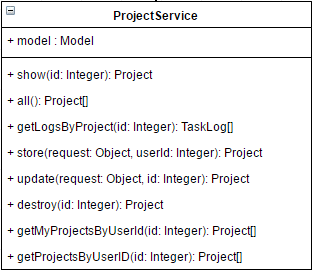
Entities

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

Method Detail

|  |  |  |  |
| --- | --- | --- | --- |
| Method name | Description | Parameters | Return |
| getUserById(id: Integer): User | This method uses for getting the id of user. | id: Integer | User Object |
| getNotifiesByUserID(id Integer):Notify[] | This method uses for getting the notifications of specific user | id: Integer | Array of Notify object |
| getMyStatistic(userId):Object | This method uses for getting their own statistics of the user | userId: Integer | Object |
| getStatisticByUserID(id: Integer): Object | This method uses for getting the statistics of specific user | Id: Integer | Object |
| getUserByCode(code: String): User | This method uses for getting the user object by the code of user | code: String | User Object |
| getMentorsByStudentId(id: Integer): User[] | This method uses for getting the list of mentors by the student ID | id: Integer | Array of User object |
| getSupervisorsByStudentId(id: Integer): User[] | This method uses for getting the list of supervisors by the student ID | id: Integer | Array of User object |
| setUserOfStudent(id: Integer, sid: Integer): User | This method uses for matching mentor and supervisor with student by inputting user ID(mentor and supervisor) and Student ID | id: Integer  sid: Integer | User object |
| deleteUserOfStudent(id: Integer, sid: Integer):User | This method uses for deleting the matching mentor and supervisor from the student object | id: Integer  sid: Integer | User object |
| create(data: Object): User | This method uses for creating the user object. | data: Object | User Object |
| createStudent(data: Object): User | This method uses for creating the student object | data: Object | User Object |
| createSupervisor(data: Object):User | This method uses for creating the supervisor object | data: Object | User Object |
| createMentor(data: Object):User | This method uses for creating the mentor object | data: Object | User Object |
| update(data: Object, id: Integer): User | This method uses for updating the user object. | data: Object  id: Integer | User Object |

### **ProjectService**



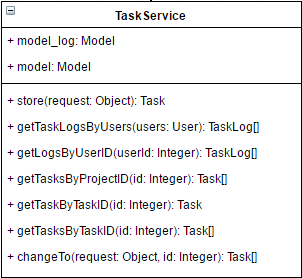
Entities

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

Method Detail

|  |  |  |  |
| --- | --- | --- | --- |
| Method name | Description | Parameters | Return |
| show(id):project | This method uses for getting a project by using project id | Id: Integer | Project object |
| all():project[] | This method uses for getting all projects | - | Array of Project object |
| getLogsByProject(id:Integer):TaskLog[] | This method uses for getting all tasklogs by the id of project | id:Integer | Array of TaskLog object |
| store(request: Object, userId: Integer): Project | This method uses for saving the input data into the project object | request: Object  userId: Integer | Project object |
| update(request: Object, id: Integer): Project | This method uses for updating the input data into the project object | request: Object  id: Integer | Project |
| destroy(id:Integer):Project | This method uses for deleting the project object | id:Integer | Project |
| getMyProjectsByUserID(id: Integer): project[] | This method uses for getting their own project objects. | id: Integer | Array of project object |
| getProjectByUserId(id: Integer): project[] | This method uses for getting all project objects by id of the user. | id: Integer | Array of project object |

### **TaskService**



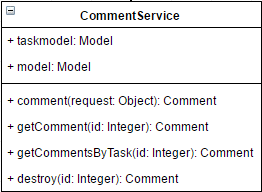
Entities

|  |  |  |  |
| --- | --- | --- | --- |
| 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. | Model |
|  | model | An object that contains the task data, provides access to that data, and implements logic to manipulate the data. | Model |

Method Detail

|  |  |  |  |
| --- | --- | --- | --- |
| Method name | Description | Parameters | Return |
| store(request: Object): Task | This method uses for saving the input data into the task object | request: Object | Task object |
| getTaskLogsByUsers(users: User): TaskLog[] | This method uses for getting all TaskLog of all related users by inputting the user object | users: User | Array of TaskLog |
| getLogsByUserID(userId: Integer): TaskLog[] | This method uses for getting all TaskLog of the specific user by inputting the user id | userId: Integer | Array of TaskLog |
| getTasksByProjectID(id: Integer): Task[] | This method uses for getting all Task of a specific project | id: Integer | Array of Task |
| getTaskByTaskID(id: Integer): Task | This method uses for getting the task by inputting task ID | id: Integer | Task object |
| getTasksByTaskID(id: Integer): Task[] | This method uses for getting all Task by inputting task ID | id: Integer | Array of Task |
| changeTo(request: Object, id: Integer): Task | This method uses for changing the status of specific task | request: Object  id: Integer | Task object |

### **CommentService**



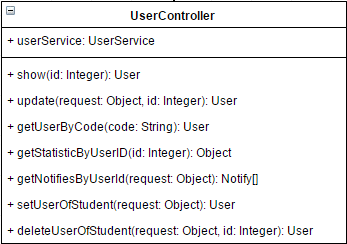
Entities

|  |  |  |  |
| --- | --- | --- | --- |
| 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 | request: Object | Comment object |
| getComment(id: Integer): Comment | This method uses for getting a comment | id: Integer | Comment object |
| getCommentsByTask(id: Integer): Comment | This method uses for getting all comment of a specific task | id: Integer | Comment object |
| destroy(id: Integer): Comment | This method uses for deleting the comment from the database | id: Integer | Comment object |

### **UserController**



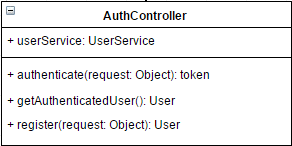
Entities

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

Method Detail

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

### **AuthController**



Entities

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

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. | request: Object | token Object |
| getAuthenticatedUser(): User | This method uses for getting the authenticated user object | - | User Object |
| 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 Object |

### **ProjectController**



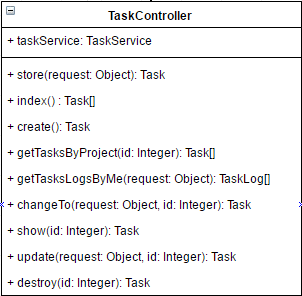
Entities

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

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. | Array of Project object |
| logs(id:Integer): TaskLog[] | This method uses for getting logs of specific project by using project id | - | Array of TaskLog object |
| store(request: Object): Project | This method uses for storing the project data which called ‘request’ object in the database. | request: Object | Project Object |
| show(id: Integer): Project | This method uses for displaying the project object data which are project name, project description, start date, image, creator name, and tasks. | id: Integer | Project Object |
| getProjectByUserID(id: Integer): Project[] | This project uses for getting the list of the projects by inputting userID | in: Integer | Array of Project object |
| myProject(id: Integer): Project[] | This project uses for getting the list of the projects of the owner. | id: Integer | Array of Project object |
| update(request: Object, id: Integer): Project | This method uses for updating the project object data. | request: Object  id: Integer | Project Object |
| destroy(id: Integer): Project | This method uses for removing the project object. | id: Integer | Project Object |

### **TaskController**



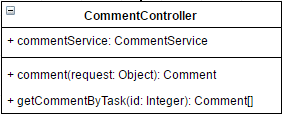
Entities

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

Method Detail

|  |  |  |  |
| --- | --- | --- | --- |
| Method name | Description | Parameters | Return |
| store(request: Object): Task | This method uses for creating a task | request: Object | Task object |
| index() : Task[] | This method uses for getting all tasks in the database | - | Array of Task object |
| getTasksByProject(id: Integer): Task[] | This method uses for getting all tasks of a specific project | id: Integer | Array of Task object |
| getTasksLogsByMe(request: Object): TaskLog[] | This method uses for getting all tasks logs of the current user | request: Object | Array of TaskLog object |
| changeTo(request: Object, id: Integer): Task | This method uses for changing the status of specific task by task id | request: Object  id: Integer | Task object |
| show(id: Integer): Task | This method uses for displaying the data of the specific task | id: Integer | Task object |
| update(request: Object, id: Integer): Task | This method uses for updating the data of a task | request: Object  id: Integer | Task object |
| destroy(id: Integer): Task | This method uses for deleting the task | id: Integer | Task object |

### **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 | request: Object | Comment object |
| getCommentByTask(id: Integer): Comment[] | This method uses for getting all comment of a specific task | id: Integer | Array of comment object |

## **Sequence diagram**

### **3.3.1 Feature 1**

1. **Register**

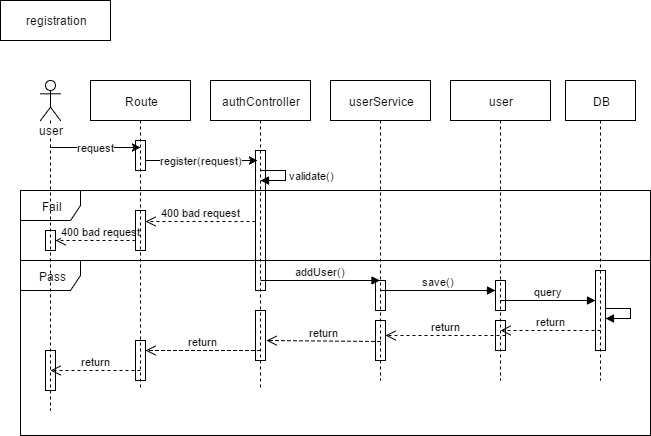


Figure 3 Sequence diagram of Registration

1. **View profile**

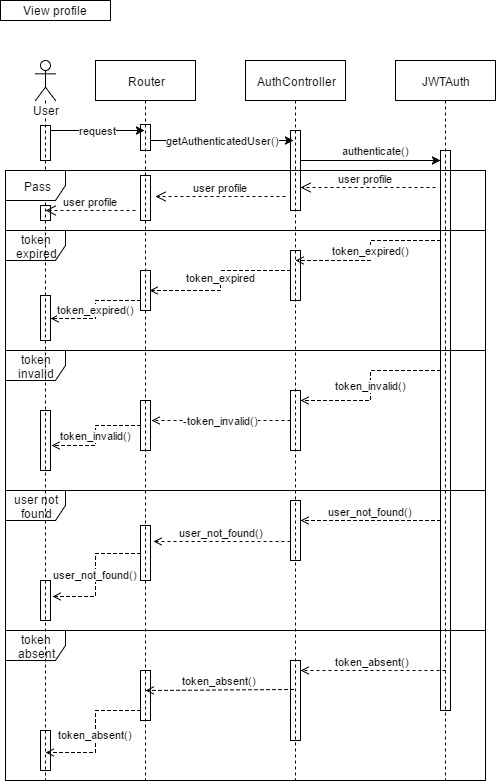


Figure Sequence diagram of View profile

1. **Edit profile**

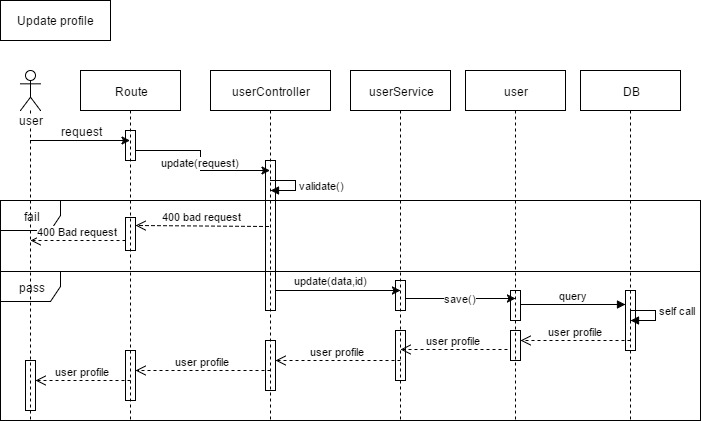


Figure 5 Sequence diagram of Edit profile

1. **Login**

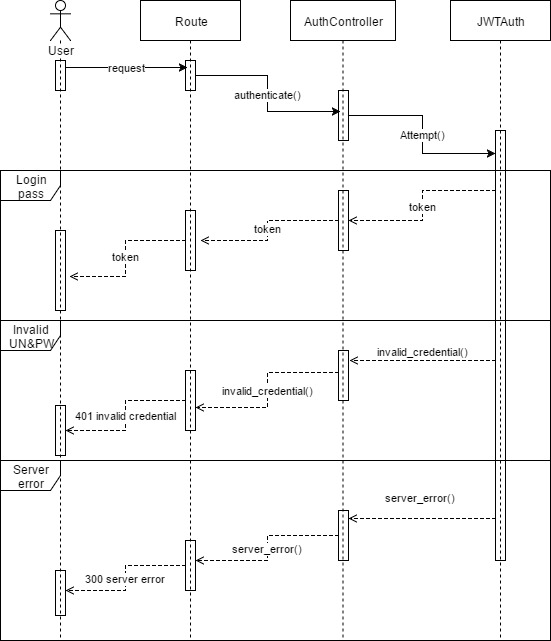


Figure 6 Sequence diagram of Login

1. **Logout**

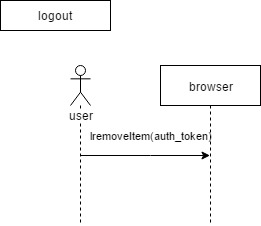


Figure 7 Sequence diagram of Logout

1. **Set code of supervisor or mentor**

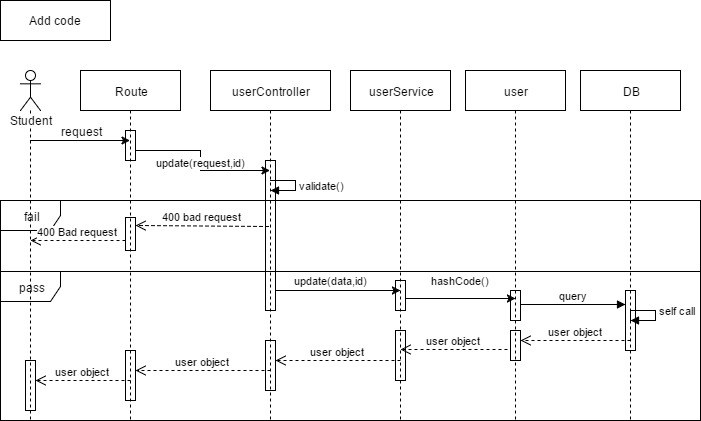


Figure 8 Sequence diagram of Set code

### **3.3.2 Feature 2**

1. **View project**

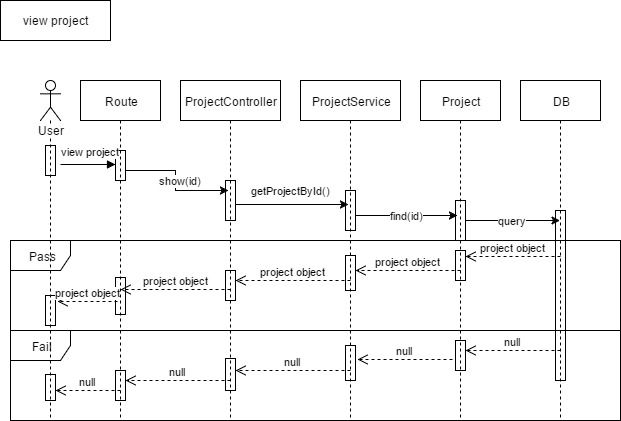


Figure 9 Sequence diagram of View project

1. **Add a project**

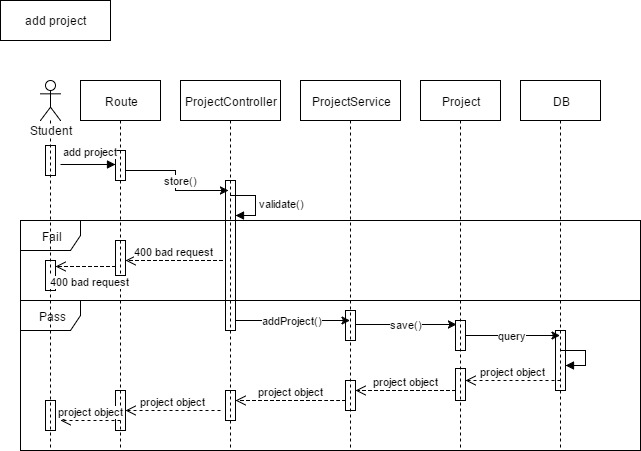


Figure 10 Sequence diagram of Add project

1. **Edit a project**



Figure 11 Sequence diagram of Edit project

1. **Delete a project**

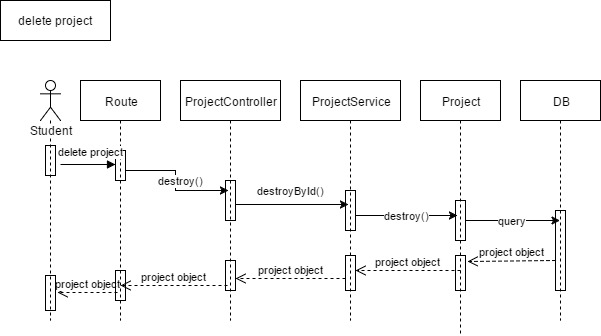


Figure 12 Sequence diagram of Delete project

1. **View a task**

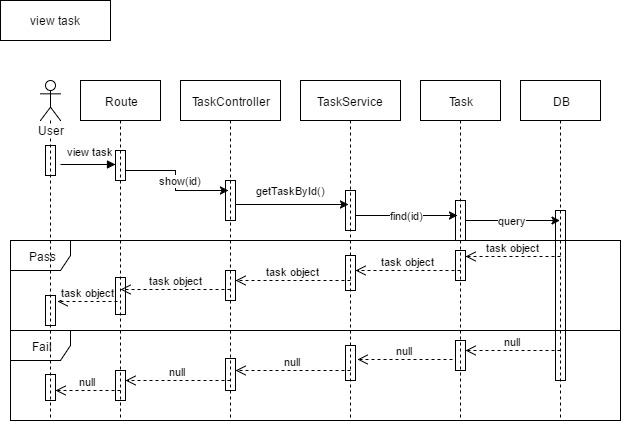


Figure 13 Sequence diagram of View task

1. **Add a task**

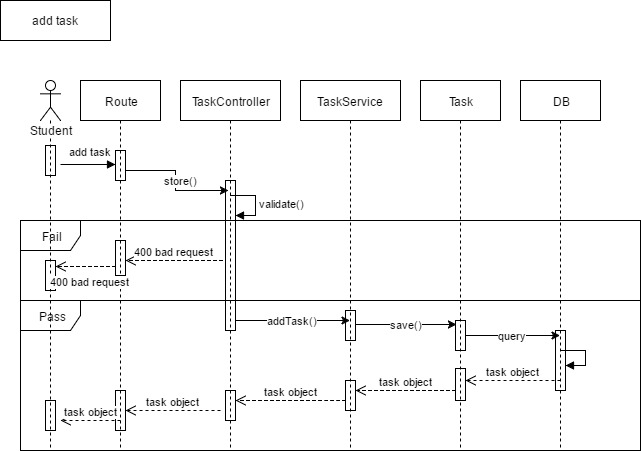


Figure 14 Sequence diagram of Add task

1. **Edit a task**

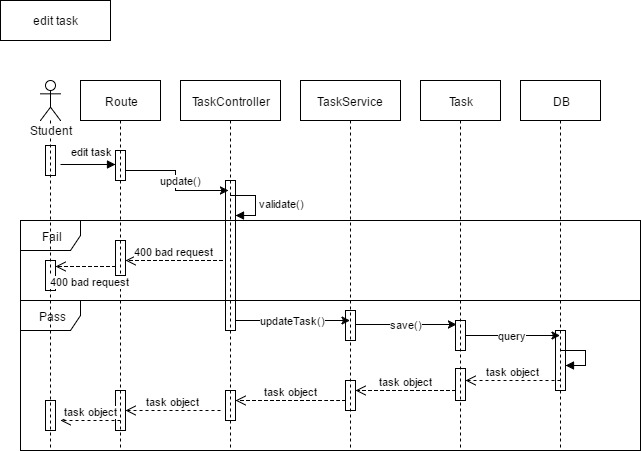


Figure 15 Sequence diagram of Edit task

1. **Delete a task**

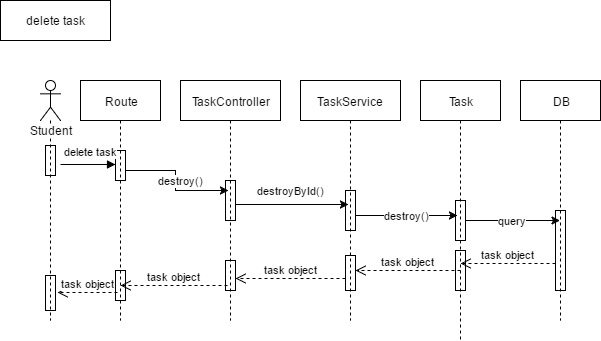


Figure 16 Sequence diagram of Delete task

1. **Move a task**

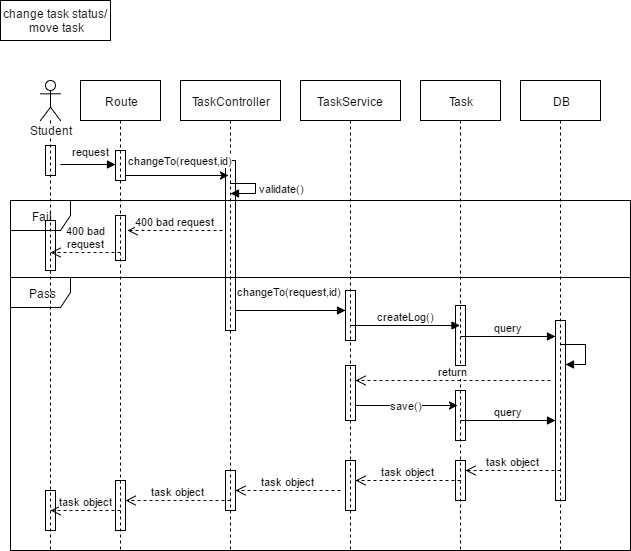


Figure Sequence diagram of Move a task

1. **View a Task log**

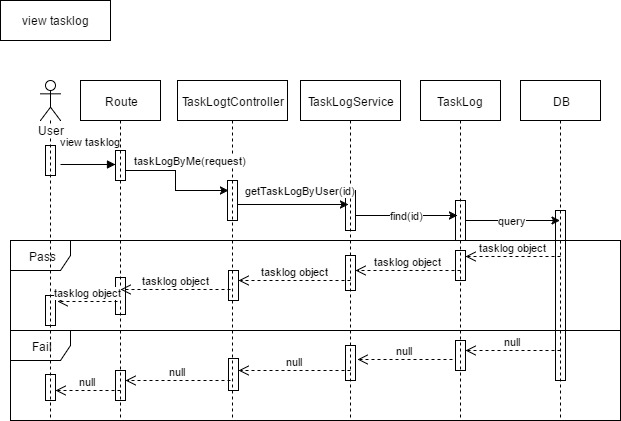


Figure Sequence diagram of View TaskLog

### 3.3.3 Feature 3

1. **View a comment**

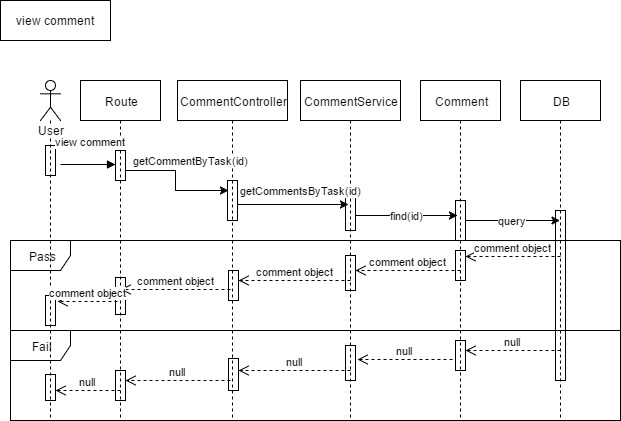


Figure Sequence diagram of View Comment

1. **Add a comment**

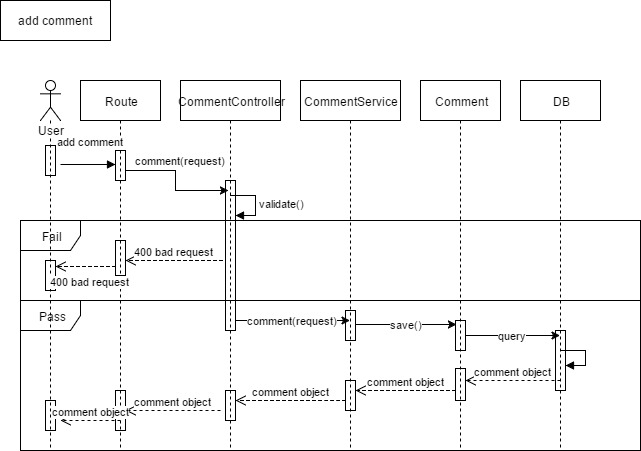


Figure Sequence diagram of Add comment

1. **Edit a comment**

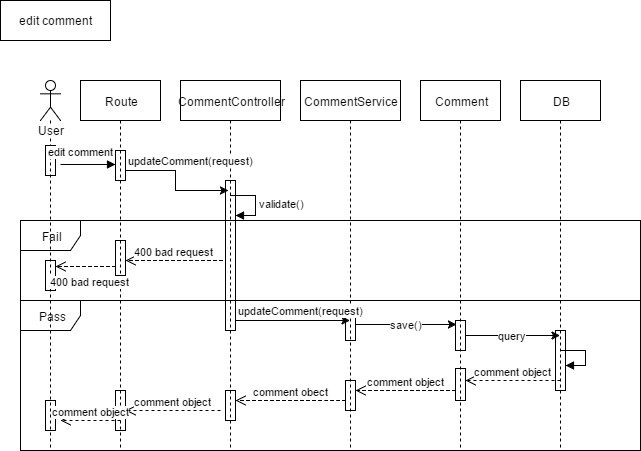


Figure Sequence diagram of Edit comment

1. **Delete a comment**

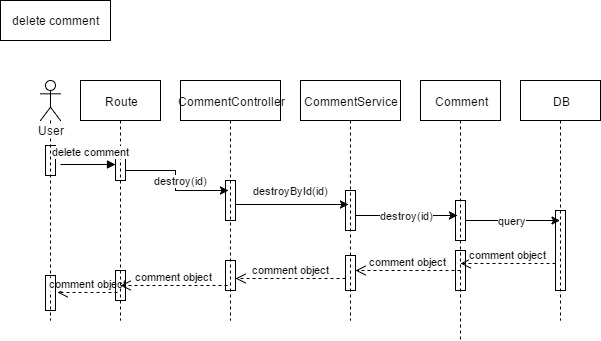


Figure Sequence diagram of Delete comment

1. **View a weekly report**

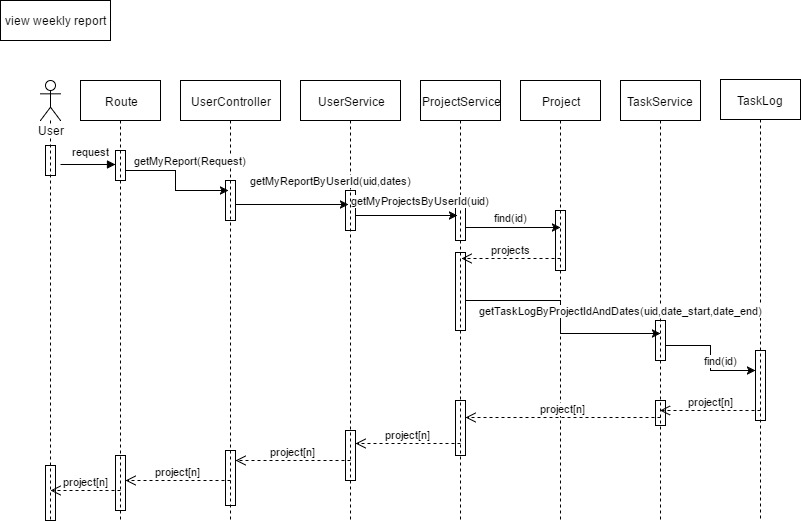


Figure Sequence diagram of View weekly report

### 3.3.4 Feature 4

1. **Receive a web notification**

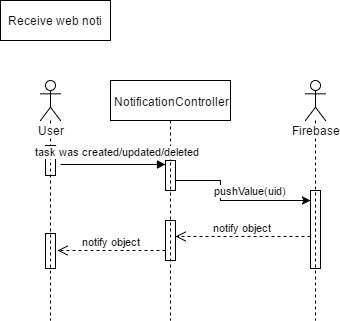


Figure Sequence diagram of Receive web notification

1. **Receive an email notification**

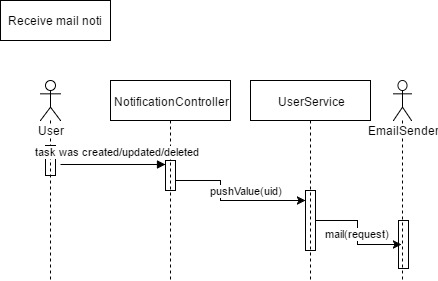


Figure Sequence diagram of Receive Email notification

# **Chapter IV | UI Design**



Figure 26 Login design

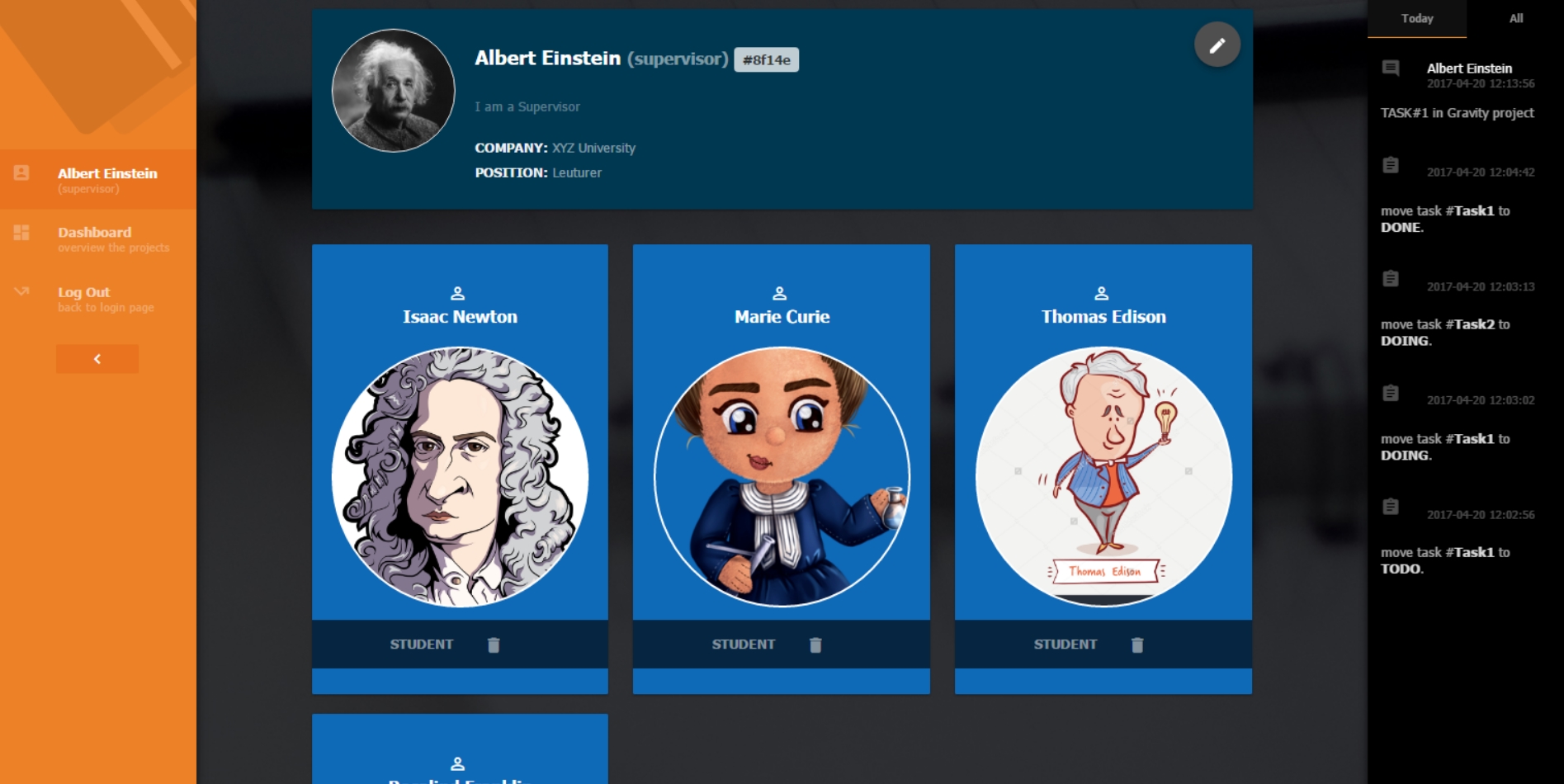


Figure Profile design

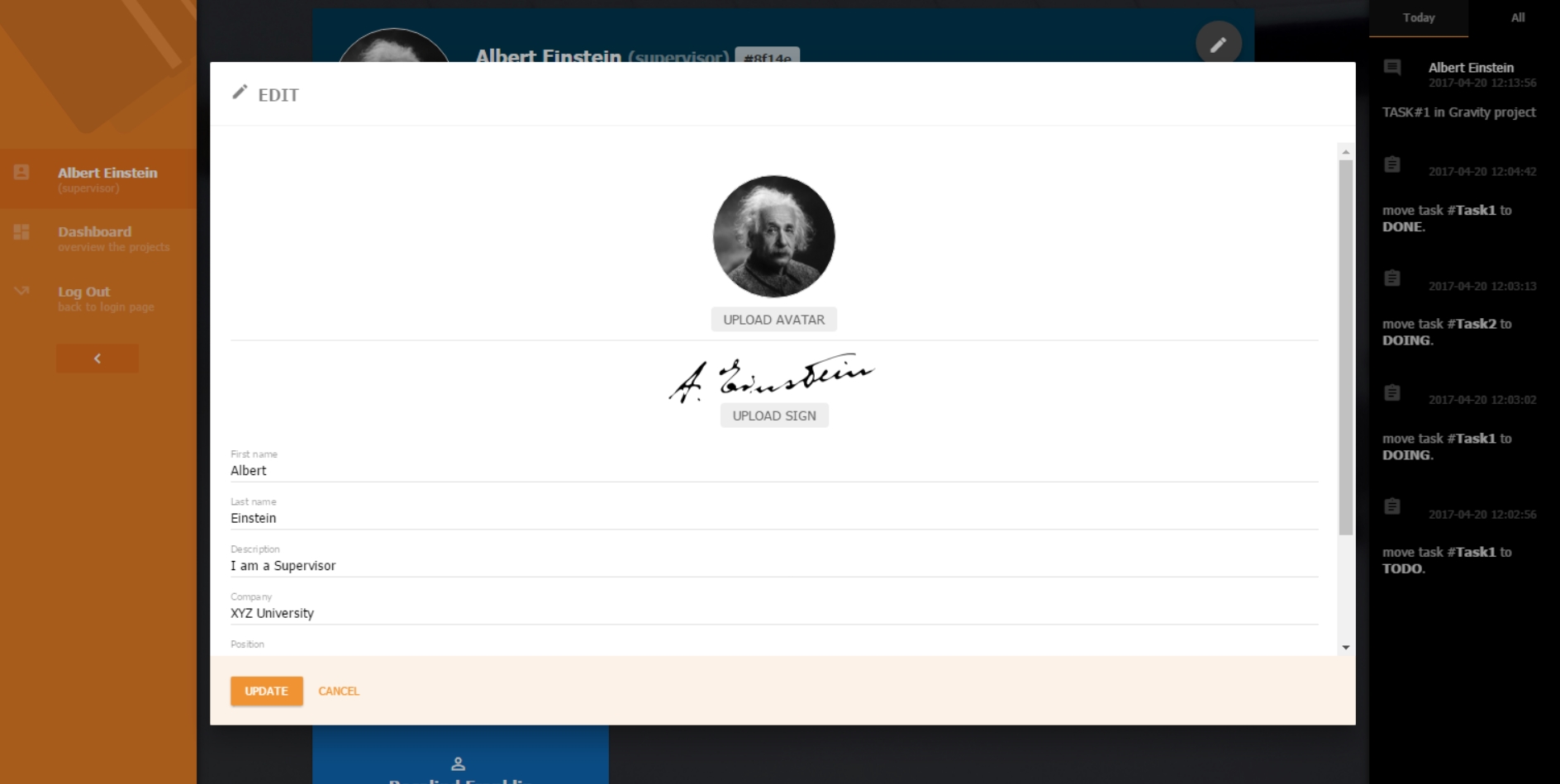


Figure 28 Edit profile design

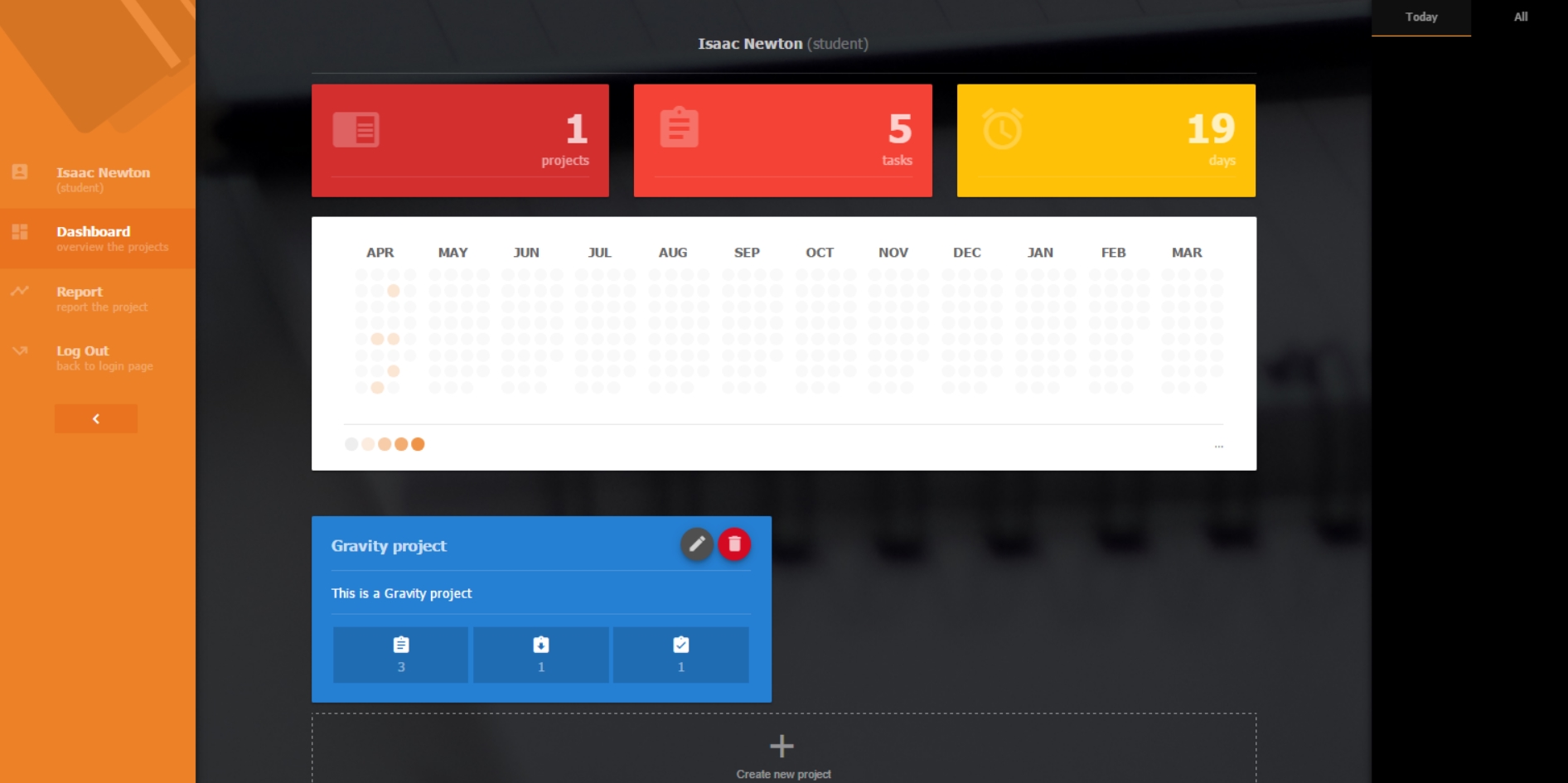


Figure 29 Student dashboard design

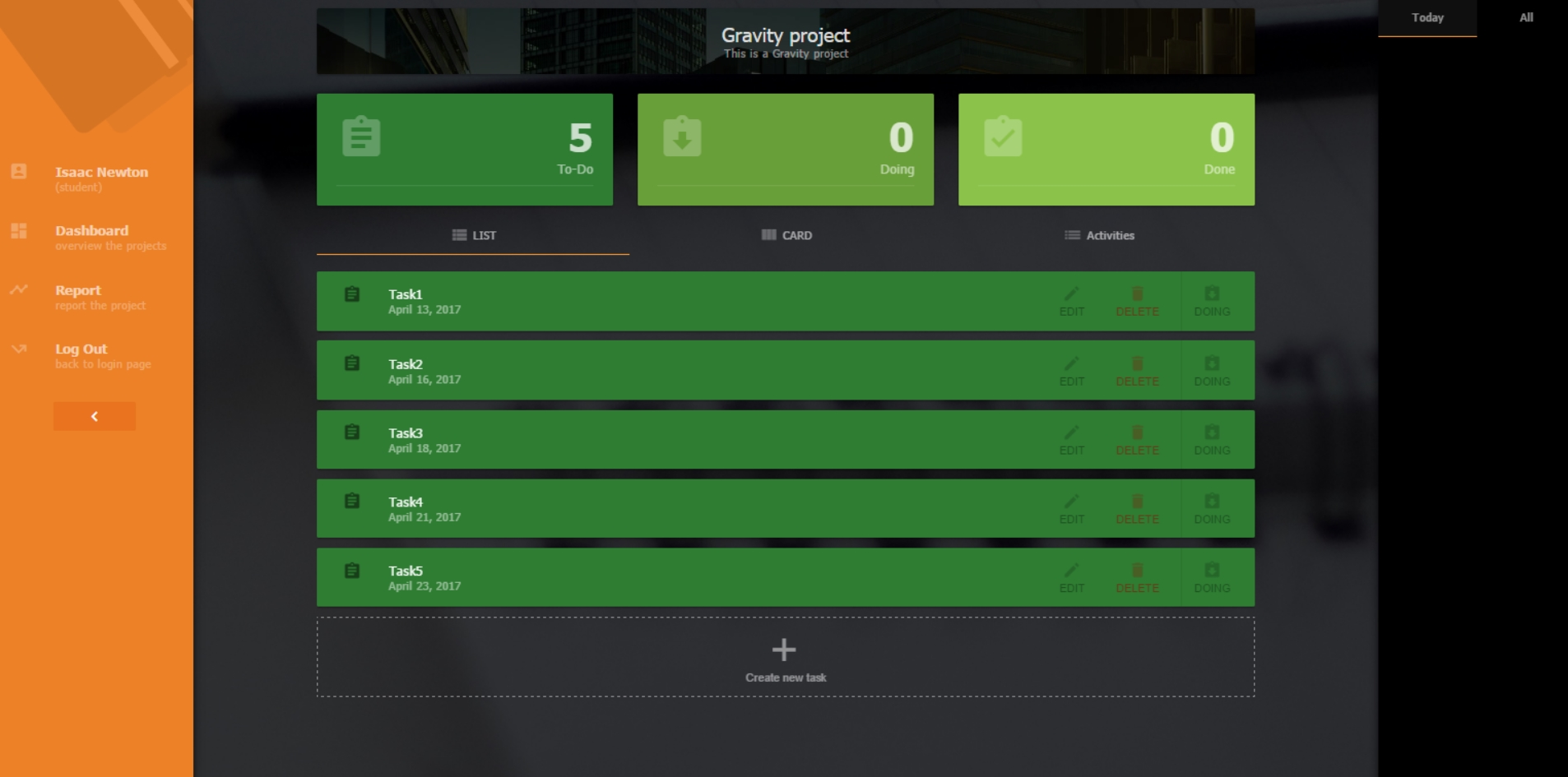


Figure 30 Task overview list design



Figure 31 Task overview card design

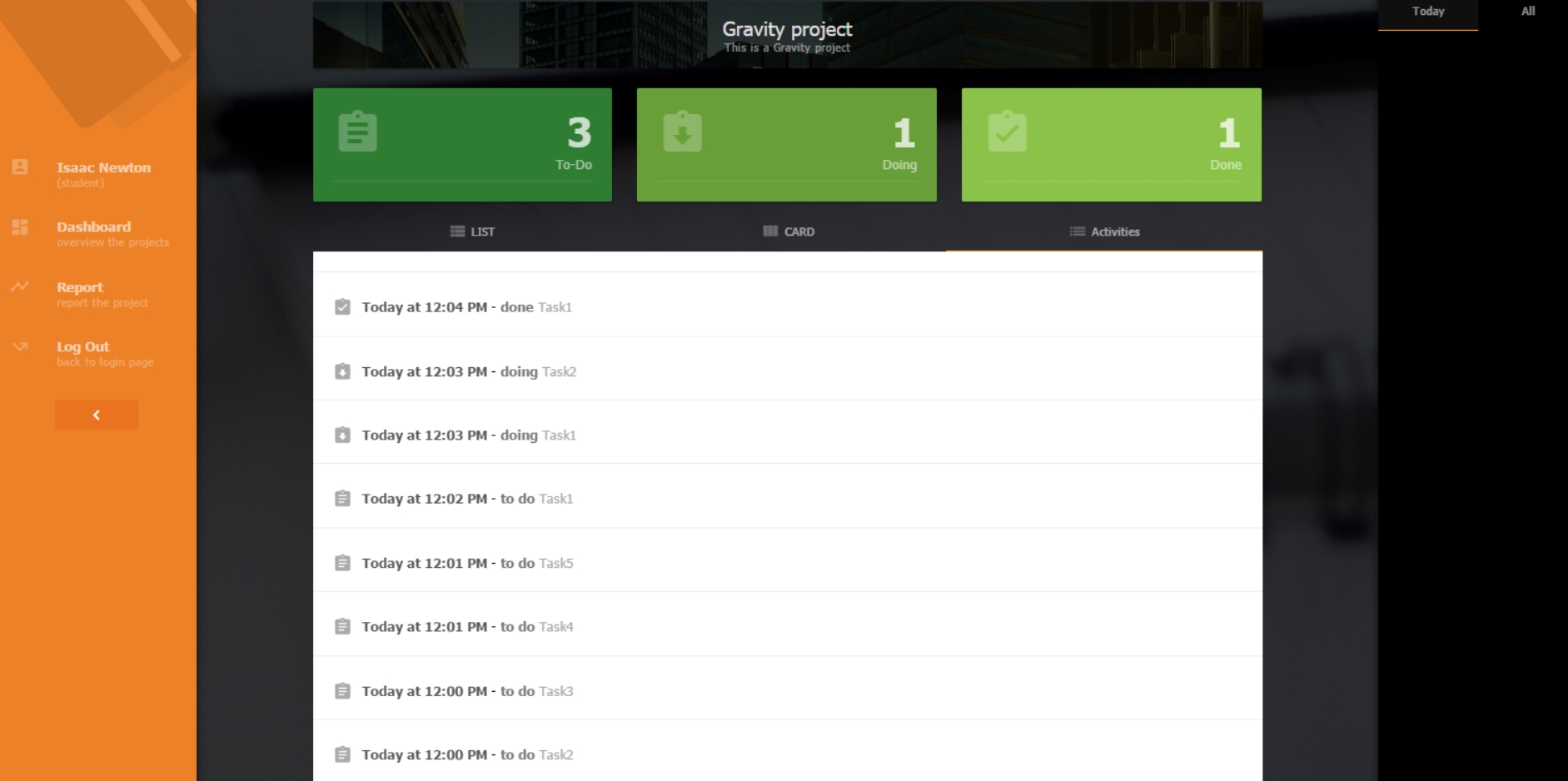


Figure Task activity overview design

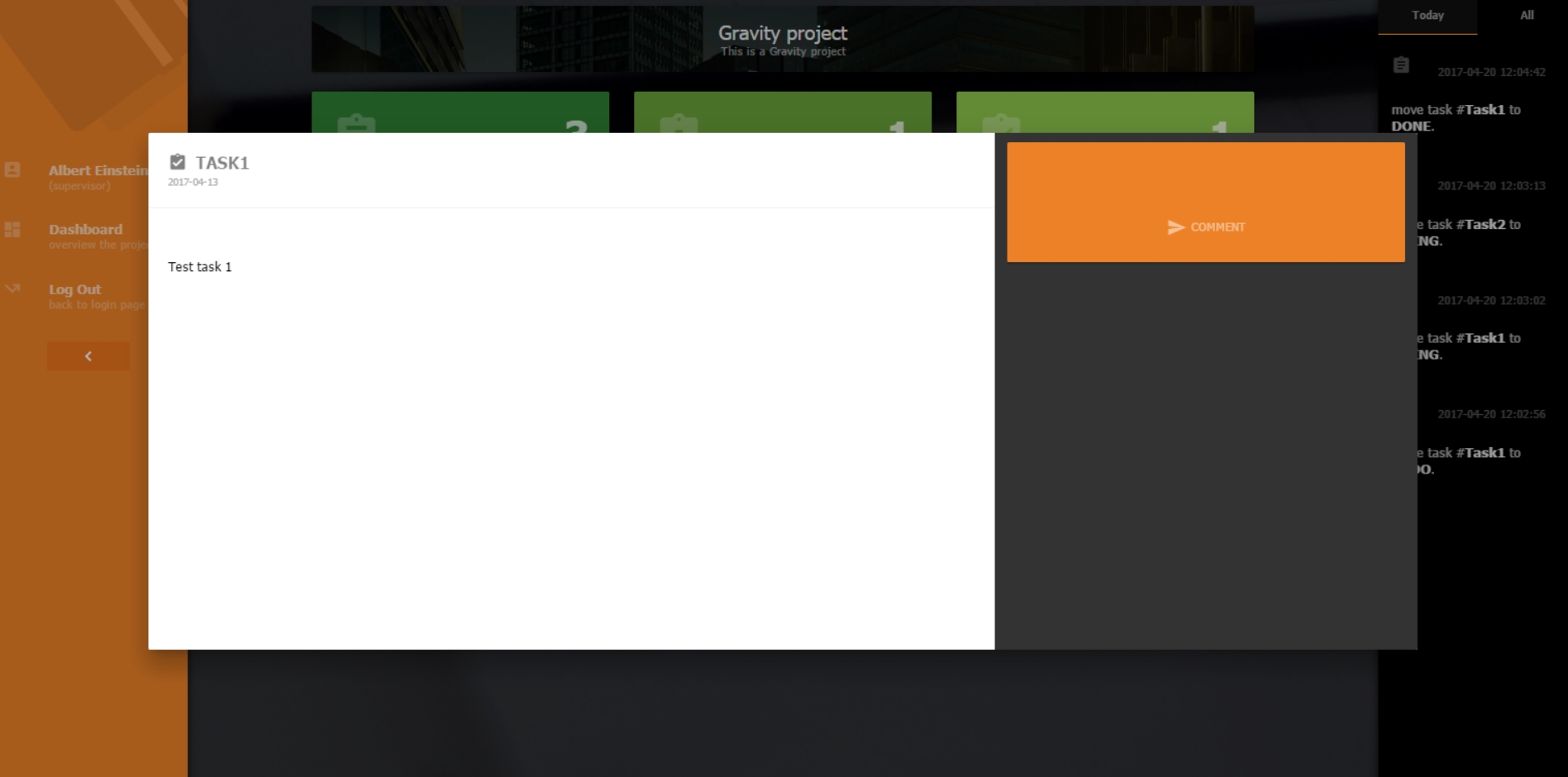


Figure Task detail design

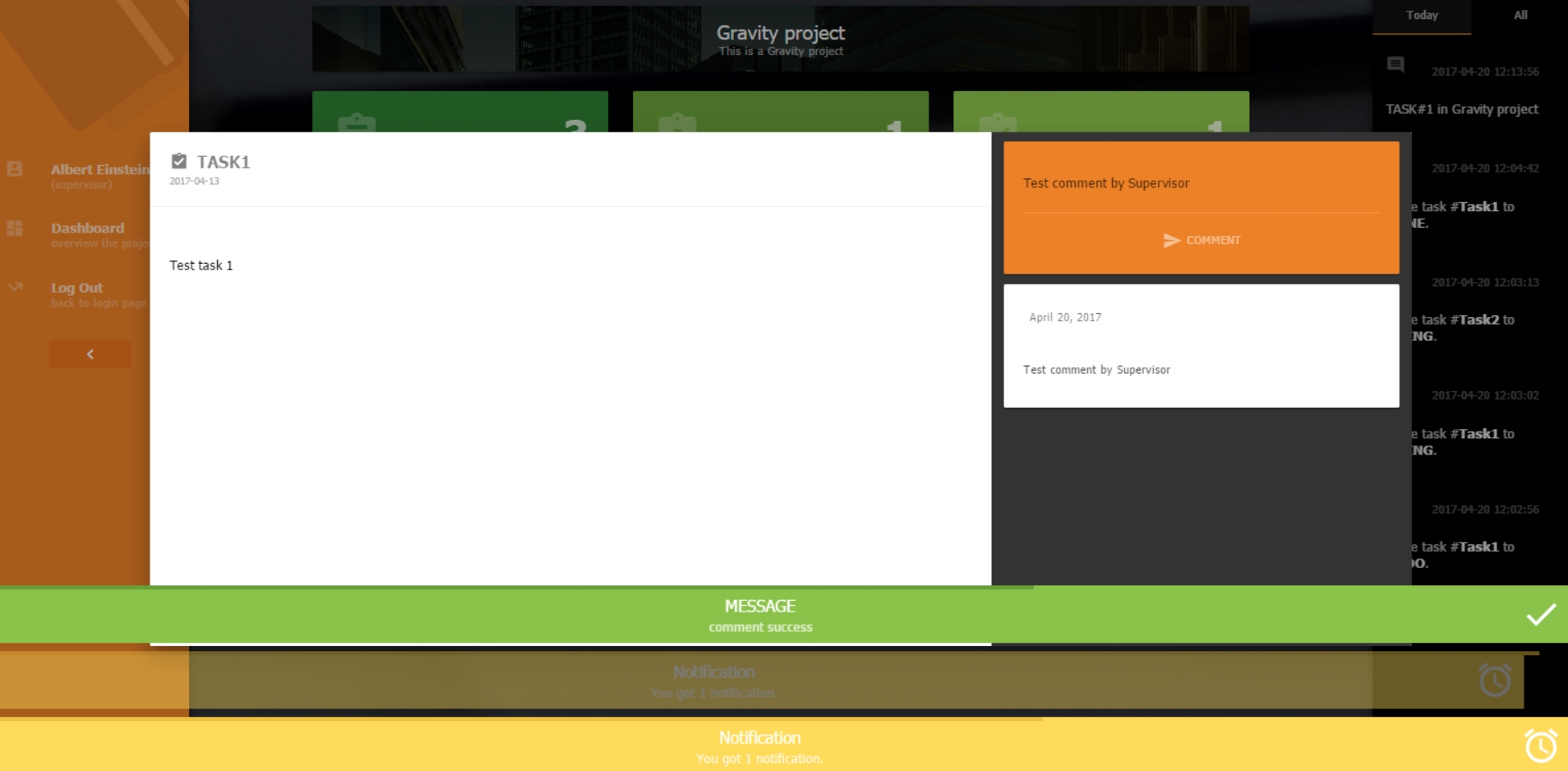


Figure Comment and notification design

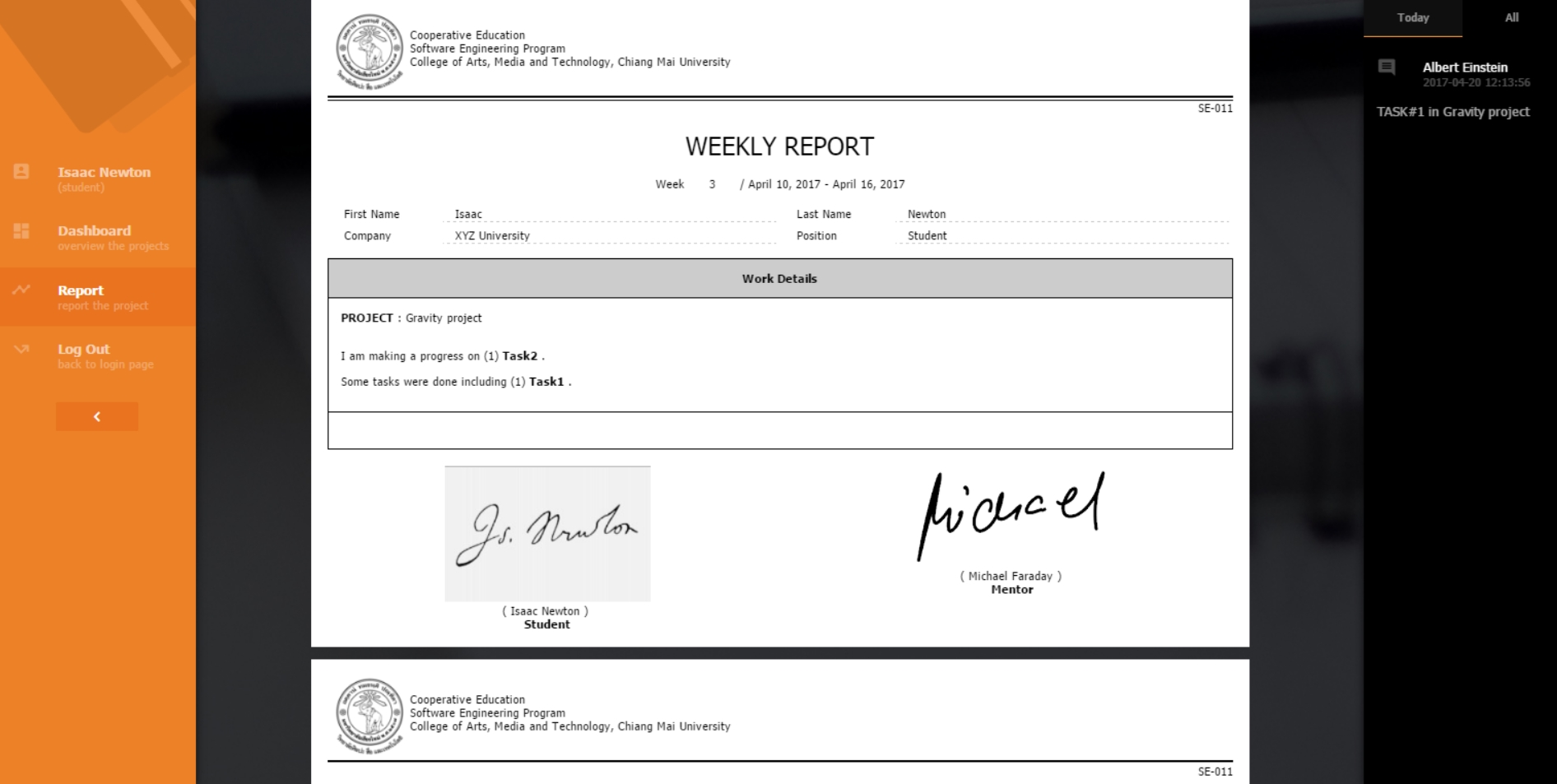


Figure Generate weekly report

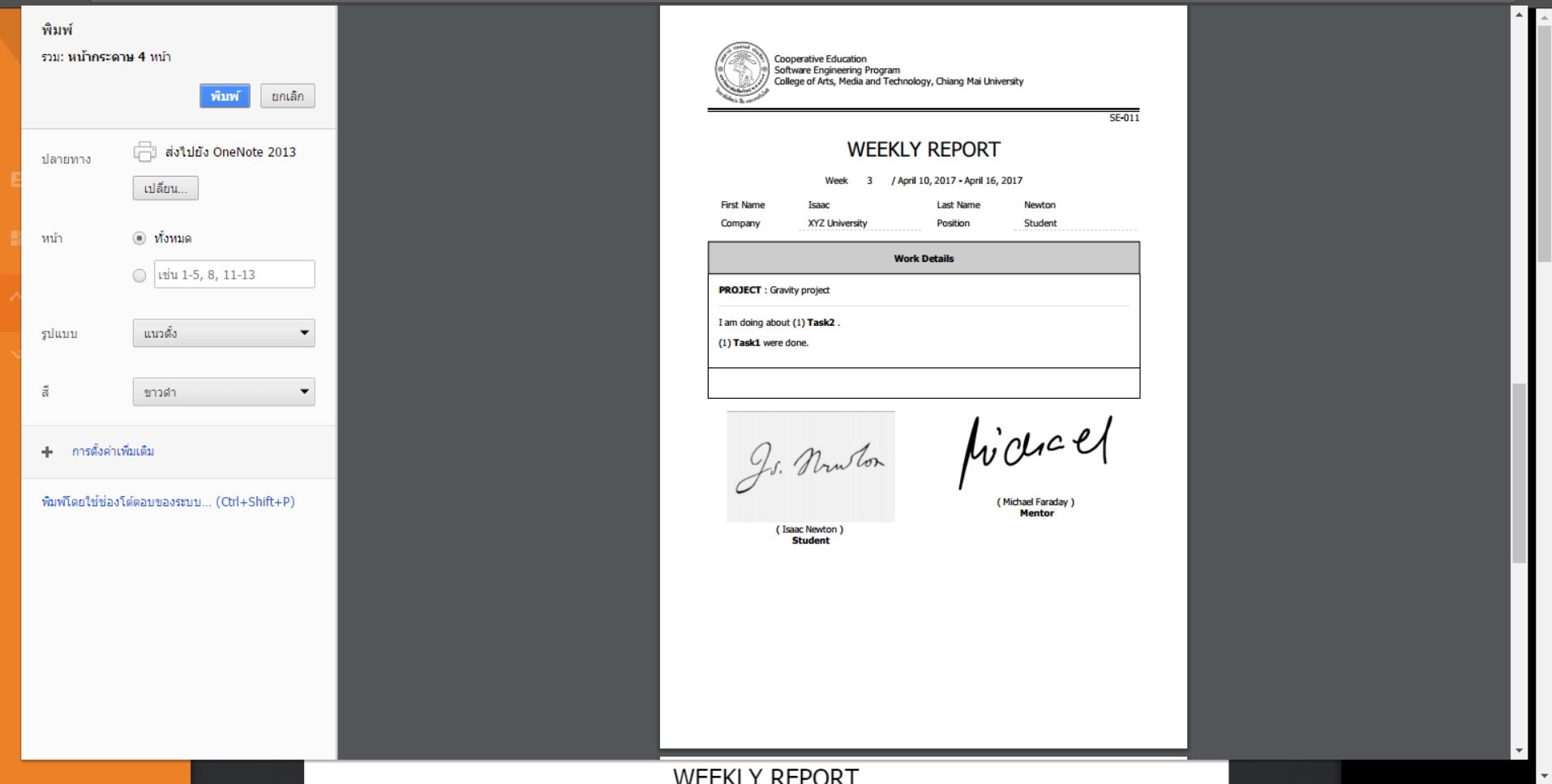


Figure Print the weekly report by browser