ID 2207 Final Project

Group 11 Zekun Du & Yuxiang Liu

User stories

App/API	Interface	Vale	Risk
Login story Login Any employee in the SEP can access the system through the login screen where she/he enters her/his user name and password. After vertifivation and based on the logged in user's authoriation level, she/he will be able to access different functionalities. Time Estimate: GUI: 0.5 hour, Logic: 2 hours		High	High
Workflow of event request			
Manage event record story SCS, FM, AM can manage the records of events, so that they can view,update, search and edit the details of some event Time estimate:GUI 0, logic: 8 hours		High	High
Search for event story split from "Manage event record " story Employee can search some event. Time estimate:GUI(already in base story), logic:(included in base story, since this is a details split story)		High	High
Search criteria story split from "Search for event" story An employee can search for a event by name or date. if no event found, she/he should see a meaningful message. Time estimate:GUI(already in base story), logic:(included in base story, since this is a details split story)		High	Medium

Manage event request story CS,SCS,AM can manage the request, so that they can do their job online Time estimate:GUI 2 hour, logic: 5 hours		High	High
Create event request story split from "Manage event request" story CS create an event request by filling in a form named "Request for Event Planning" Time estimate:GUI(already in base story), logic:(included in base story, since this is a details split story)		High	Medium
Forward/reject event request story split from "Manage event request" story SCS can make decision on the event request screen by "Forword" button and "Reject" button. Time estimate:GUI(already in base story), logic:(included in base story, since this is a details split story)		High	Low
Update event request story split from "Manage event request" story FM can add feedback and update the event request. Time estimate:GUI(already in base story), logic:(included in base story, since this is a details split story)		High	Medium
Approve/reject event request story split from "Manage event request" story Approve/reject event request ADmanager can make the decision on the event request screen by "Forword" button and "Reject" button. Time estimate:GUI(already in base story), logic:(included in base story, since this is a details split story)		High	Low

Approve/reject criteria story split from "Approve/reject event request " story Approve/reject criteria After AM makes the decision about the request, the SCSM should be able to see a meaningful message. Time estimate:GUI(already in base story), logic:(included in base story, since this is a details split story)		High	Low
Update event detail story split from "Manage event record" story SCSM can update the event detail got from the business meeting to the system by the event screen. Time estimate:GUI(already in base story), logic:(included in base story, since this is a details split story)		High	Medium
Staff recuritment management			
Manage HR request story Production/Service manager and HRManager can manager the HR request by the system Time estimate:GUI 0, logic: 8 hours		High	High
Create HR request story split from "Manage HR request " story Production/Service manager can create the HR request and send it to the HR Assistant. Time estimate:GUI(already in base story), logic:(included in base story, since this is a details split story)		High	High
Manage staff recruitment story HR manager can manange all the things which are related to the staff recruiment Time estimate:GUI 1.5 hour, logic: 4 hours		High	High

Manage candidate record story split from "Manage staff recuriment" story HR Manager and HR Assistant can view, update the canditates record. Time estimate:GUI(already in base story), logic:(included in base story, since this is a details split story)		Medium	High
View candidate personal record details story split from "Manage candidate record" story HR Manager and HR Assistant can view the personal details of any canditate. Time estimate:GUI(already in base story), logic:(included in base story, since this is a details split story)		Medium	Medium
Update candidate record story split from "Manage candidate record" story HR can update the details of any candidate and add the decision to the record. Decision inludes "approve, reject and suspend" Time estimate:GUI(already in base story), logic:(included in base story, since this is a details split story)		Medium	Medium
View candidate interviews history story split from "Manage candidate record" story HR can view the interviews history which some candidate has attended to. Time estimate:GUI(already in base story), logic:(included in base story, since this is a details split story)		Low	Medium
Workflow of task distribution to S/P department			
Manage application story Production/Service manager can manage an application with the client needs from his department, and send tasks to each sub-team. Time estimate:GUI 2 hour, logic: 8 hours		High	High

Initiate application story split from "Manage application" story Production/Service manager can fill an application.Include a list of tasks and other details about the application. Time estimate:GUI(already in base story), logic:(included in base story, since this is a details split story)		High	High
View application story split from "Manage application" story Production/Service manager can view the details of an application . Time estimate:GUI(already in base story), logic:(included in base story, since this is a details split story)		Medium	High
Manage tasks story split from "Manage application" story Production/Service manager can create tasks and check the comments of them. And sub-team can check and edit the task. Time estimate:GUI(already in base story), logic:(included in base story, since this is a details split story)		High	Medium
Send tasks story split from "Manage tasks " story Production/Service manager can send the tasks to any subteam which belongs to his/her department. Time estimate:GUI(already in base story), logic:(included in base story, since this is a details split story)		Medium	Medium
Manage tasks record story split from "Manage tasks " story Production/Service manager and the leader of sub-team can search, view and update the tasks which is related to them through the tasks screen. Time estimate:GUI(already in base story), logic:(included in base story, since this is a details split story)		Medium	Medium

View task details story split from "Manage tasks record " story Production/Service manager and the leader of sub-team can view details of the tasks which are related to them Time estimate:GUI(already in base story), logic:(included in base story, since this is a details split story)		Medium	Low
View task comments story split from "View task details" story Production/Service manager can view the comment. Time estimate:GUI(already in base story), logic:(included in base story, since this is a details split story)		Medium	Medium
View tasks history story split from "Manage tasks record " story Production/Service manager and the leader of sub-team can view the task history. Time estimate:GUI(already in base story), logic:(included in base story, since this is a details split story)		Low	High
Edit task story split from "Manage tasks record " story Sub-team can edit the task, add a plan and add the comment in case of extra budget needed. Time estimate:GUI(already in base story), logic:(included in base story, since this is a details split story)		Medium	Low
Create plan story split from "Edit task story " story Sub-team can create a task plan and add it the task record. Time estimate:GUI(already in base story), logic:(included in base story, since this is a details split story)		Medium	Medium
Add comment story split from "Edit task story " story Sub-team can add comment asking for extra budget. Time estimate:GUI(already in base story), logic:(included in base story, since this is a details split story)		Low	Low

Financial requests management			
Manage financial request story Production/Service manager and Financial manager can manage the financial request . Time estimate:GUI 0, logic: 8 hours		High	High
Create financial request story split from "Manage financial request" story Production/Service manager can fill in a form and create a financial request, and send it to the Finacial manager. Time estimate:GUI(already in base story), logic:(included in base story, since this is a details split story)		High	Medium
Manage financial request record story split from "Manage financial request" story All the financial request must be stored in the system, Production/Service manager can view the requests, and the Financial manager can view,update financial request . Time estimate:GUI(already in base story), logic:(included in base story, since this is a details split story)		High	High
View financial request record details story split from"Manage financial request record" story The Financial manager can view, update financial request. Time estimate:GUI(already in base story), logic:(included in base story, since this is a details split story)		High	Medium
Create financial task story split from"Manage financial request record" story The Financial manager can create a financial task and send it to accountant. Time estimate:GUI(already in base story), logic:(included in base story, since this is a details split story)		Medium	High

View financial task story split from"Manage financial request record" story The accountant can view the financial request sent by Financial Manager. Time estimate:GUI(already in base story), logic:(included in base story, since this is a details split story)		Medium	Low
Finalize financial request story split from "Manage financial request" story the Financial manager can update the negotiation record to the financial request and finalizes the financial request. Also, Production/Service manager should be able to see a meaningful message. Time estimate:GUI(already in base story), logic:(included in base story, since this is a details split story)		High	Low

Release Planning

	High Value	Medium Value	Low Value
High Risk	11	3	1
Medium Risk	7	6	1
Low Risk	4	3	1

Iteration plan

According to the sorting combinations of value and risk. We focus on the combinations of (high, high), (high, medium), (high, low), (medium, high) and (medium, medium) and choose them for the first release.

	Stories to be implemented
First iteration (7/10/2017-9/10/2017)	Search for event, Search criteria, create event request, Forward/reject event request, Update event request, Approve/reject event request, Approve/reject criteria, Update event detail, Manage event request, Manage event record, Login.
Second iteration (10/10/2017-12/10/2017)	Create HR request, Manage HR request, Manage staff recruitment, Manage candidate record, View candidate personal record details, Update candidate record, Initiate application, View application, Send tasks, Manage tasks record, View task details.
Third iteration (13/10/2017-15/10/2017)	View task comments, View tasks history, Edit task, Create plan, Add comment, Manage tasks, Manage application, Manage financial request, Create financial request, Manage financial request record, Create financial task, View financial task, Finalize financial request

The metaphor

Metaphor	System
Resturant	Swedish Event Planning Company.
Waiter	Customer Servie officer who creates event request for client.
Head waiter	Senior customer service officer
Casher	Financial Manager
Boss	Administration Manager
Order	Event Request
Dishes	Tasks
Chef	Subteam
Head Chef	Service/Production Manager
Recruiting chef	Staff recruitment
Recipe	Task plan
Rising price for dishes	Financial request

Test-driven pair programming description

This project source codes include two main kind of classes: actor classes and request/task classes.

Actor classes describe employees, for example, financial manager. This class has functionality to access information and take actions to make business flow.

Request/task classes describe information, requests and tasks that are created and handled by actor classes. They contain the main business details of the system.

There will be two example (HR manager class & HR request class).

First let's talk about HR request class. Only if this class is finalized, the classes that use it will be finalized.

1. Write a test:

```
void test_hr_request_class(){
   hr_request * hr_r = new hr_request("Oct. 13", "Service",
"sep123", 2, "Those who can cook.");
   hr_r->add_hire("lucy");
   hr_r->view_hr_request();
}
```

2. Implement just enough to compile

```
class hr_request{
public:
   hr_request(string dat, string depart, string pr, int n,
string req);
   void add_hire(string name);
   void view_hr_request();
};
```

3. Implement just enough to run

```
class hr_request{
string date;
    string department;
    string project_reference;
    int req_number;
    string recruitment_requirement;
public:
   string hire_list[10];
hr_request(string dat, string depart, string pr, int n,
string req){
           date = dat;
        department = depart;
        project_reference = pr;
        req_number = n;
        recruitment_requirement = req;
}
void add_hire(string name) {
        int i = 0;
        while (hire_list[i] != "") {
            i++;
        }
        hire_list[i] = name;
}
void view_hr_request(){
        cout << "\n>>Date: " << get_date() << endl;
        cout << ">>Department: " << get_department() << endl;
        cout << ">>Project reference: " <<
get_project_reference() << endl;
        cout << ">>Required recruitment number: " <<
get_number() << endl;
        cout << ">>Recruitment requirements: " <<
get_recruitment_requirement() << endl;
        cout << ">>Hire: ";
        for (int i = 0; hire_list[i] != ""; i++){
            cout << hire_list[i] << " ";
        }
        cout << endl;
    }
};
```

4. Think other needed attributes and needed fuctions then refine.

```
class hr_request{
    string date;
    string department;
    string project_reference;
    int req_number;
    string recruitment_requirement;
    string send_to_hr;
    string status;
public:
    string hire_list[10];
    hr_request(){
        req_number = 0;
        status = "unsettled";
    }
    hr_request(string dat, string depart, string pr, int n,
string req){
        date = dat;
        department = depart;
        project_reference = pr;
        req_number = n;
        recruitment_requirement = req;
        status = "unsettled";
        send_to_hr = "";
    }
    void change_date(string d){date = d;}
    string get_date(){return date;}
    void change_department(string d){department = d;}
    string get_department(){return department;}
    void change_number(int n){req_number = n;}
    int get_number(){return req_number;}
    void change_project_reference(string
pr){project_reference = pr;}
    string get_project_reference(){return project_reference;}
    void change_recruitment_requirement(string
s){recruitment_requirement = s;}
```

```
    string get_recruitment_requirement(){return
recruitment_requirement;}
    void change_status(string s){status = s;}
    string get_status(){return status;}
    void change_send_to_hr(string h){send_to_hr = h;}
    string get_send_to_hr(){return send_to_hr;}
    void add_hire(string name){
        int i = 0;
        while (hire_list[i] != "") {
            if (i >= 10){
                cout << "full! "<< endl;
                return;
            }
            i++;
        }
        hire_list[i] = name;
    }
    void view_hr_request(){
        cout << "\n>>Date: " << get_date() << endl;
        cout << ">>Department: " << get_department() << endl;
        cout << ">>Project reference: " <<
get_project_reference() << endl;
        cout << ">>Required recruitment number: " <<
get_number() << endl;
        cout << ">>Recruitment requirements: " <<
get_recruitment_requirement() << endl;
        cout << ">>Status: " << get_status() << endl;
        cout << ">>Send to (hr): " << get_send_to_hr() <<
endl;
        cout << ">>Hire: ";
        for (int i = 0; hire_list[i] != ""; i++){
            cout << hire_list[i] << " ";
        }
        cout << endl;
    }
};
```

5. After refining, I need some external functions and variables to support this class.

```
hr_request * hr_request_list[hr_request_number];

hr_request * get_hr_request(string pr, string depart){
    for (int i = 0; hr_request_list[i] != NULL; i++){
        if (hr_request_list[i]->get_project_reference() ==
pr
            && hr_request_list[i]->get_department() ==
depart)
            return hr_request_list[i];
    }
    return NULL;
}

void view_hr_request(string pr, string depart){
    hr_request * hr_req;
    bool flag = false;
    for (int i = 0; hr_request_list[i] != NULL; i++){
        flag = true;
        hr_req = hr_request_list[i];
        if (pr != "all"){
            if (hr_req->get_project_reference() != pr){
                flag = false;
            }
        }
        if (depart != "all"){
            if (hr_req->get_department() != depart){
                flag = false;
            }
        }
        if (flag == true){
            hr_req->view_hr_request();
        }
    }
}

void add_to_hr_request_list(hr_request * hr){
    int i = 0;
    while (hr_request_list[i] != NULL) i++;
    if (i >= hr_request_number){
        cout << "HR request full!" << endl;
        return;
    }
    hr_request_list[i] = hr;
    return;
}

void view_hr_request_hr(string st){
    hr_request * hr_req;
    bool flag = false;
    for (int i = 0; hr_request_list[i] != NULL; i++){
        flag = true;
        hr_req = hr_request_list[i];
        if (hr_req->get_send_to_hr() != st)
            flag = false;
        if (flag == true){
            hr_req->view_hr_request();
        }
    }
}

string resume_list[10];

void init_resume(){
    resume_list[0] = "\nlucy green\nAAA University BCs\n/---
details---/\n";
    resume_list[1] = "vivian black\nBBB University BCs\n/---
details---/\n";
    resume_list[2] = "swan wu\nCCC University BCs\n/---
details---/\n";
```

```
    resume_list[4] = "alan stone\nDDD University BCs\n/---
details---/\n";
}
```

Second, the HR manager class.

1. Write a test.

```
void test_hr_manager_class(){
   hrm h1(“simon”, “a0003”, “HR Manager”);
   h1. view_hr_req();
   h1. order_task(“sep123”, “Service”, “maria”);
}
```

2. Implement just enough to compile

```
class hrm {
public:
    hrm(string n, string ID, string p);
    void view_hr_req();
    void order_task(string pr, string depart, string st);
};
```

3. Implement just enough to run

```
class hrm {
   string name;
    string position;
    string id;
public:
    hrm(string n, string ID, string p){
        name = n;
        position = p;
        id = ID;
    }
    void view_hr_req(){
        view_hr_request("all", "all");
    }
    void order_task(string pr, string depart, string st){
        hr_request * p;
        if (p = get_hr_request(pr, depart))
            p->change_send_to_hr(st);
    }
};
```

4. Fulfill the functionality and refine.

   In this part, we use an super class called employee. It has basic functions to store “name”, “position” and “ID”, the HR manager class add additional functions to employee class.

```
class hrm : employee{
public:
    hrm();
    hrm(string n, string ID, string p){
        name = n;
        position = p;
        id = ID;
        add_to_list();
    }
    void view_hr_req(){
        view_hr_request("all", "all");
    }
    void order_task(string pr, string depart, string st){
        hr_request * p;
        if (p = get_hr_request(pr, depart))
            p->change_send_to_hr(st);
    }
};
```

Acceptance tests

Hi, welcome to read this acceptance tests for SEP project.

Actually it is a guide book, and it will have a whole picture of what SEP work.

The programmer uses the Code::Blocks, so he press "ctrl+F9" to compile and then press "ctrl+F10" to run.

Let's talk about user story first.

1. Login (later login part will be ignored)

   Other users login part will not show in this file but they exist.

   show: Choose an actor: <Name>

   enter: sarah

   show: Please enter your username:

   enter: sarah

   show: Please enter your password:

   enter: sarah123

   show: Login success!

2. Event request

   show: sarah wants to...<action>

   enter: ls

   show: >>create: create a event request

   >>view: view all related event request status

   enter: create

   show: Event request name:

   enter: Bob's Marriage

   show: Date:

   enter: Oct. 13

   show: Project reference:

   enter: sep001

   show: Details:

enter: Bob will have a marriage in Stockholm, with 40 guests, on Nov. 1.

show: Choose an actor: <Name>

enter: sarah

show: sarah wants to...<action>

enter: view

show:

>>Event request name: Bob's Marriage

>>Project reference: sep001

>>Date: Oct. 13

>>Created by: sarah

>>Detail: Bob will have a marriage in Stockholm, with 40 guests, on Nov. 1.

>>SCS decision: unsettled

>>FM feedback:

>>AM decision: unsettled

show:

Choose an actor: <Name>

enter: janet

show: janet wants to...<action>

enter: appr

show: Project reference:

enter: sep001

show:

Choose an actor: <Name>

enter: alice

show: alice wants to...<action>

enter: ls

show:

view-f: view financial request

view-e: view event request

create-t: create financial task

add-e: add event request feedback

add-f: add financial request feedback

solve-f: solve financial request

show:

Choose an actor: <Name>

enter: alice

show: alice wants to...<action>

enter: view-e

show:

>>Event request name: Bob's Marriage

>>Project reference: sep001

>>Date: Oct. 13

>>Created by: sarah

>>Detail: Bob will have a marriage in Stockholm, with 40 guests, on Nov. 1.

>>SCS decision: Approve

>>FM feedback:

>>AM decision: unsettled

show:

Choose an actor: <Name>

enter: alice

show: alice wants to...<action>

enter: add-e

show: Project reference:

enter: sep001

show: Feedback:

enter: This may need 10,000kr for budget. It's beneficial for our company.

show:

Choose an actor: <Name>

enter: mike

show: mike wants to...<action>

enter: view

show:

>>Event request name: Bob's Marriage

>>Project reference: sep001

>>Date: Oct. 13

>>Created by: sarah

>>Detail: Bob will have a marriage in Stockholm, with 40 guests, on Nov. 1.

>>SCS decision: Approve

>>FM feedback: This may need 10,000kr for budget. It's beneficial for our company.

>>AM decision: unsettled

show:

Choose an actor: <Name>

enter: mike

show: mike wants to...<action>

enter: appr

show: Project reference:

enter: sep001

show:

Choose an actor: <Name>

enter: janet

show: janet wants to...<action>

enter: view-2

show:

>>Event request name: Bob's Marriage

>>Project reference: sep001

>>Date: Oct. 13

>>Created by: sarah

>>Detail: Bob will have a marriage in Stockholm, with 40 guests, on Nov. 1.

>>SCS decision: Approve

>>FM feedback: This may need 10,000kr for budget. It's beneficial for our company.

>>AM decision: Approve

3. HR request

show: Choose an actor: <Name>

enter: jack

show: jack wants to...<action>

enter: ls

show:

create-f: create a new financial request

view-f: view financial request status

create-hr: create an hr request

view-hr: view all hr request created by jack

view-one: view one specific hr request by project reference

create-t: create a new subteam task

view-t: view production tasks

view-p: view production subteam task plans

appr: approve a task plan and give feedback

rej: reject a task plan and give feedback

enter: create-hr

show: Date:

enter: Oct. 13

show: Project reference:

enter: sep001

show: Recruitment number:

enter: 1

show: Recruitment requirement:

enter: Who can do audio job.

show:

Choose an actor: <Name>

enter: simon

show: simon wants to...<action>

enter: ls

show:

view: view all hr request

order: order a hr request to an hr assistant

view

show:

>>Date: Oct. 13

>>Department: Production

>>Project reference: sep001

>>Required recruitment number: 1

>>Recruitment requirements: Who can do audio job.

>>Status: unsettled

>>Send to (hr):

>>Hire:

show:

Choose an actor: <Name>

enter: simon

show: simon wants to...<action>

enter: order

show: Project reference:

enter: sep001

show: Department:

enter: Production

show: Send to:

enter: Maria

show: Choose an actor: <Name>

enter: maria

show: maria wants to...<action>

enter: ls

show:

view-all: view all hr requests sent to her

view-r: view all resumes

hire: recruit a person

enter: view-all

show:

>>Date: Oct. 13

>>Department: Production

>>Project reference: sep001

>>Required recruitment number: 1

>>Recruitment requirements: Who can do audio job.

>>Status: unsettled

>>Send to (hr): maria

>>Hire:

show:

Choose an actor: <Name>

enter: maria

show: maria wants to...<action>

enter: view-r

show:

lucy green

AAA University BCs

/---details---/

vivian black

BBB University BCs

/---details---/

swan wu

CCC University BCs

/---details---/

show:

Choose an actor: <Name>

enter: maria

show: maria wants to...<action>

enter: hire

show: Project reference:

enter: sep001

show: Department:

enter: Production

show: Hire:

enter: lucy

show:

Choose an actor: <Name>

enter: jack

show: jack wants to...<action>

enter: view-hr

show:

>>Date: Oct. 13

>>Department: Production

>>Project reference: sep001

>>Required recruitment number: 1

>>Recruitment requirements: Who can do audio job.

>>Status: settled

>>Send to (hr): maria

>>Hire: lucy

4. Production / service task

show: Choose an actor: <Name>

enter: jack

show: jack wants to...<action>

enter: create-t

show: Date:

enter: Oct.14

show: Project reference:

enter: sep001

show: Send to: <name>

enter: tobias

show: Task details:

enter: /****detail*****/

show: Choose an actor: <Name>

enter: tobias

show: tobias wants to...<action>

enter: view-t

show:

//Production subteam member tobias views subteam tasks

>>Task Send Date: Oct.14

>>Department: Production

>>Project Reference: sep001

>>Detail: /****detail*****/

>>Task Send to: tobias

show:

Choose an actor: <Name>

enter: tobias

show: tobias wants to...<action>

enter: ls

show:

>>create: create a new task plan

>>view-t: view task

>>view-p: view task plan (including feedback)

enter: create

show: Date:

enter: Oct.14

show: Department:

enter: Production

show: Project reference:

enter: sep001

show: Task plan detail:

/***plan**detail***/

show: Extra budget:

enter: 2000

show:

Choose an actor: <Name>

enter: jack

show: jack wants to...<action>

enter: view-p

show:

//Production manager jack views subteam task PLANs

>>Creater name: tobias

>>Task Send Date: Oct.14

>>Department: Production

>>Project Reference: sep001

>>Detail: /***plan**detail***/

>>Extra_budget: 2000

>>Feedback:

show:

Choose an actor: <Name>

enter: jack

show: jack wants to...<action>

enter: ls

show:

create-f: create a new financial request

view-f: view financial request status

create-hr: create an hr request

view-hr: view all hr request created by jack

view-one: view one specific hr request by project reference

create-t: create a new subteam task

view-t: view production tasks

view-p: view production subteam task plans

appr: approve a task plan and give feedback

rej: reject a task plan and give feedback

enter: appr

show: Please enter project reference: sep001

show: Please enter creator name: tobias

show: Please enter feedback: Good.

show: Choose an actor: <Name>

enter: tobias

show: tobias wants to...<action>

enter: view-p

show:

//Production subteam member tobias views subteam tasks

>>Creater name: tobias

>>Task Send Date: Oct.14

>>Department: Production

>>Project Reference: sep001

>>Detail: /***plan**detail***/

>>Extra_budget: 2000

>>Feedback: Approved! Good.

5. Financial request

show: Choose an actor: <Name>

enter: jack

show: jack wants to...<action>

enter: create-f

show: Date: Oct. 14

show: Project reference: sep001

show: Amount: 2000

show: Reason: Extra equipments

show: Choose an actor: <Name>

enter: alice

show: alice wants to...<action>

enter: view-f

show:

//Financial manager alice views financial requests statuses.

>>Reqest Date: Oct. 14

>>Request Department: Production

>>Project Reference: sep001

>>Requested Amount: 2000kr

>>Reason: Extra equipments

>>Status: unsolved

>>Feedback From Financial Manager:

show:

Choose an actor: <Name>

enter: alice

show: alice wants to...<action>

enter: ls

show:

view-f: view financial request

view-e: view event request

create-t: create financial task

add-e: add event request feedback

add-f: add financial request feedback

solve-f: solve financial request

enter: create-t

show: Project reference: sep001

show: Choose an actor: <Name>

enter: sophia

show: sophia wants to...<action>

enter: view

show:

//Financial accountant sophia views financial tasks.

>>Reqest Date: Oct. 14

>>Request Department: Production

>>Project Reference: sep001

>>Requested Amount: 2000kr

>>Reason: Extra equipments

>>Comments From Financial Manager:

show: Choose an actor: <Name>

enter: alice

show: alice wants to...<action>

enter: add-f

show: Enter project reference:

enter: sep001

show: Feedback:

enter: We think your request is reasonable

show: Choose an actor: <Name>

enter: alice

show: alice wants to...<action>

enter: solve-f

show: Enter project reference: sep001

show: Choose an actor: <Name>

enter: jack

show: jack wants to...<action>

enter: view-f

show:

//Production manager jack views financial requests status

>>Reqest Date: Oct. 14

>>Request Department: Production

>>Project Reference: sep001

>>Requested Amount: 2000kr

```
>>Reason: Extra equipments
>>Status: Solved
>>Feedback From Financial Manager:  We think your request is reasonable
```

6. Extra user story: check record

show: Choose an actor: <Name>

enter: record

show:

```
/- alice, Financial Manager -/
Project reference: sep001; Roll: Financial Request Receiver
Project reference: sep001; Roll: Financial Task Sender

/- fredrik, Accountant -/
Project reference: sep001; Roll: Financial Task Receiver

/- sophia, Accountant -/
Project reference: sep001; Roll: Financial Task Receiver

/- jack, Production Manager -/
Project reference: sep001; Roll: Subteam Task Sender
Project reference: sep001; Roll: Financial Request Sender

/- natalie, Service Manager -/

/- tobias, Production Team -/
Project reference: sep001; Roll: Subteam Task Receiver

/- julia, Production Team -/
```

/- helen, Service Team -/

/- kate, Service Team -/

/- mike, Administration Manager -/

/- sarah, Customer Service -/

/- sam, Customer Service -/

/- janet, Senior Customer Service -/

/- simon, HR Manager -/

/- maria, HR -/

7. Exit user story

   show: Choose an actor: <Name>

   enter: exit

   show:

   Process returned 0 (0x0)   execution time : 671.000 s

   Press any key to continue.

Feedback

In the previous assignments, we followed the object-oriented analysis and design approach. This time, we applied XP to our final project. This first method we applied tries to define all the requirements in the project initiation stage. After that, if there are some changes of customer requirements, it will be a disaster because it may need us to change most of the jobs we did before. Instead of trying to avoid changes, XP intends to improve responsiveness to changing customer requirements. To adopt new customer requirements, it introduces several releases and splits it into multiple iterations. In addition, test-driven pair programming and intensive code review make it easier to validate our design.