**1. Formulera minst 5 krav från beskrivningen som use cases eller user stories.**

**1:**

Title: **Create user**

Actor: Admin

Scope: “Organization of tasks” system

Postcondition:

- User is added to the database.

Preconditions:

- A valid API-key is needed.

Trigger: The user invokes a create user request in the system.

Basic flow:

1. The Admin use postman to create user.

2. Admin chooses POST command.

3. Admin write in first name, last name and username and then click send.

Extensions:

A. If username is less then 10 character, show error message.

B. If all the names are not written, show error message.  
 C. If the API-key is invalid show error message.

**2:**

Title: **Deactivate user**

Scope: “Organisation of tasks” system

Actor: User

Postcondition:

- User has been deactivated.

Preconditions:

- User has to have status active.

- User id must exist in db with unique number.

Trigger: The Admin invokes a deactivate user request in the system.

Basic flow:

1. User choses deactivate User button in the system.

2. User is presented with a list of users in the system.

3. User selects a User from the list or uses the search box for user id.

4. To deactivate the User the button for deactivate user is checked.

5. The information is saved with confirm button.

Extensions:

A. User name does not exist in database.

B. The user has already been inactivated.

**3:**

Title: **Add user to team**

Scope: “Organization of tasks” system

Actor: User

Postcondition:  
 - User is added to team

Preconditions:

- User exists and is not already part of a team.  
 - Team exists and does not contain 10 or more users.

Trigger: The user invokes a add user to team request in the system.

Basic flow:

1. User selects the add user to team function.   
 2. User specifies which user to add to which team.  
 3. The selected user is added to the selected team.

Extensions:

A. If user don’t exist show error message.  
 B. If team has 10 or more members show error message.

C. If team don’t exist show error message.

**4:**

Title: **Assign Work item to user**

Scope: “Organization of tasks” system

Actor: User

Postcondition:

- Work item added to user.  
 - Users list of work items is increased by 1.

Preconditions:

- The user which the user is trying to assign a work item to exists.  
 - The work item the user is trying to assign to a user exists.

Trigger: The Admin invokes a assign work item to user request in the system.

Basic flow:

1. User selects assign work item to user function in the system.   
 2. User specifies what work item to assign to which user.  
 3. Work item is assigned to user.

Extensions:

A. If user already has more then 5 work items assigned show error message

**5:**

Title: **Create Issue and assign it to WorkItem**

Scope: “Organization of tasks” system

Actor: User

Postcondition:

- Issue is created.

- When a Issue is added to a work item, status for work item is changed to Unstarted.

Preconditions:  
 - the work item user is trying to assign the issue to exists.

- An Issue can only be added to work item that has status Done.

Trigger: The user invokes a create issue request in the system.

Basic flow:

1. User selects create issue and assign it to work item function in the system.   
 2. User writes a issue description and selects which work item to assign it to.  
 3. Issue is created and assigned to work item.

Extensions:  
 A. If user tries to add the issue to a work item with any other status then “Done” show   
 error message.

B. If work item don’t exist show error message.

**2. Formulera minst 3 icke-funktionella krav för systemet.**

**1:** The database should be able to handle 10.000 requests at the same time.

**2:** The response time should never be longer than 5 seconds.

**3:**  There must be support on weekdays between 08-16.

**3. Skriv testfall för samtliga era krav enligt ovan.**

**1001 Create user with valid input:**

● Go to site [http://](http://demo.guru99.com)localhost:8080/users

● Enter first name

● Enter surname

● Enter username

● Click Submit Test data:

● Admin password = password Password = password

**Expected results:**

● User should be created and added to the database.

Actual results: Pass / Fail:

● As Expected: **Pass**

**1002 Create user with invalid input:**

● Go to site [http://](http://demo.guru99.com)localhost:8080/users

● Enter first name

● Enter surname

● Enter username

● Click Submit Test data:

● Admin password = pass123 Password = pass123

**Expected results:**

● User should not be created.

Actual results: Pass / Fail:

● As Expected: **Pass**

**1003 Create user with invalid username-length:**

● Go to site [http://](http://demo.guru99.com)localhost:8080/users

● Enter first name

● Enter surname

● Enter username

● Click Submit Test data:

● Admin password = password Password = password

**Expected results:**

● User should not be created.

Actual results: Pass / Fail:

● As Expected: **Pass**

**1004 Inactivate user:**

● Go to site [http://](http://demo.guru99.com)localhost:8080/users/1002/inactivate

● Click Submit Test data:

**Expected results:**

● User should be set to active: 0 and all the users work items should be set to unstarted.

Actual results: Pass / Fail:

● As Expected: **Pass**

**1005 Inactivate user that doesn’t exist:**

● Go to site [http://](http://demo.guru99.com)localhost:8080/users/9999/inactivate

● Click Submit Test data:

**Expected results:**

● Return message user could not be found.

Actual results: Pass / Fail:

● As Expected: **Fail**

**1006 Add User to team:**

● Go to site [http://](http://demo.guru99.com)localhost:8080/teams/1/users/1002

● Click Submit Test data:

**Expected results:**

● User 1002 should be added to team 1

Actual results: Pass / Fail:

● As Expected: **Pass**

**1007 Add User to team that doesn’t exist:**

● Go to site [http://](http://demo.guru99.com)localhost:8080/teams/9/users/1002

● Click Submit Test data:

**Expected results:**

● User 1002 should not be added to team 9

Actual results: Pass / Fail:

● As Expected: **Pass**

**1008 Add a non-existent user to a team:**

● Go to site [http://](http://demo.guru99.com)localhost:8080/teams/1/users/1099

● Click Submit Test data:

**Expected results:**

● User 1099 should not be added to team 1

Actual results: Pass / Fail:

● As Expected: **Pass**

**1009 Assign a work item to a user:**

● Go to site [http://](http://demo.guru99.com)localhost:8080/workitems/1/users/1001

● Click Submit Test data:

**Expected results:**

● Work item should be assigned a user

Actual results: Pass / Fail:

● As Expected: **Pass**

**1010 Assign a non-existent work item to a user:**

● Go to site [http://](http://demo.guru99.com)localhost:8080/workitems/6/users/1001

● Click Submit Test data:

**Expected results:**

● User should not be assigned to a work item that doesn’t exist.

Actual results: Pass / Fail:

● As Expected: **Pass**

**1011 Assign a work item to a non-existent user:**

● Go to site [http://](http://demo.guru99.com)localhost:8080/workitems/1/users/1098

● Click Submit Test data:

**Expected results:**

● Work item should not be assigned a user.

Actual results: Pass / Fail:

● As Expected: **Pass  
  
1012 Create and add an issue to a work item.**

● Go to site [http://](http://demo.guru99.com)localhost:8080/workitems/1/issues

● Click Submit Test data:

**Expected results:**

● Creates and adds an issue to a specific work item.

Actual results: Pass / Fail:

● As Expected: **Pass**

**1013 Add an issue to a non-existent work item.**

● Go to site [http://](http://demo.guru99.com)localhost:8080/workitems/99/issues

● Click Submit Test data:

**Expected results:**

● An issue should not be created and added to a non-existent work item.

Actual results: Pass / Fail:

● As Expected: **Pass**

**4. Testa systemet enligt testfallen.**

**1001 Create user with valid input and admin password (token key):**

Expected result: 201 created.

Actual result: 201 created. **Test passed.**

**1002 Create user with invalid password (token key):**

Expected result: Status 401 unauthorized.

Actual result: Status 401 unauthorized. **Test passed.**

**1003 Create user with valid admin password & invalid username-length:**

Expected result: Status 400 bad request.

Actual result: Status 400 bad request, username cannot be shorter than 10 characters. **Test passed.**

**1004 Deactivate user:**

Expected result: 200 OK.

Actual result: 200 OK. **Test passed.**

**1005 Deactivate user that doesn’t exist:**

Expected result: Message from application why user was not deactivated.

Actual result: Status 404 not found. **Test failed.**

**1006 Add User to team:**

Expected result: Status 200 OK, user added to team.

Actual result: Status 200 OK. **Test passed.**

**1007 Add User to team that doesn’t exist:**

Expected result: Status 400 bad request, error message: no team matching that id was found.

Actual result: Status 400 bad request, error message: no team matching that id was found. **Test passed.**

**1008 Add a non-existent user to a team:**

Expected result: Status 400 bad request, error message: no user matching that id was found.

Actual result: Status 400 bad request, error message: no user matching that id was found. **Test passed.**

**1009 Assign a work item to a user:**

Expected result: Status 200 OK, work item was assigned to user.

Actual result: Status 200 OK, work item was assigned to user. **Test passed.**

**1010 Assign a non-existent work item to a user:**

Expected result: Status 400 bad request, error message: no work item with that id.

Actual result: Status 400 bad request, error message: no work item with that id. **Test passed.**

**1011 Assign a work item to a non-existent user:**

Expected result: Status 400 bad request, error message: no user with that id.

Actual result: Status 400 bad request, error message: no user with that id. **Test passed.**

**1012 Create and add an issue to a work item.**Expected result: Status 201 Created, issue created and added to workitem with provided id.

Actual result: Status 201 Created, issue created and added to workitem with provided id. **Test passed.**

**1013 Add an issue to a non-existent work item.**Expected result: Status 400 bad request, error message: No workitem was found with id: X".  
Actual result: Status 400 bad request, error message: No workitem was found with id: X". **Test Passed**

**Instruktioner:**

* Ha mySQL redo med server connection
* Ha Postman redo också
* Kör jar-filen i Terminalen eller välj ett simplare alternativ och öppna zip-filen och bygg upp allt i din intellij IDEA
* Nu kan du börja använda Postman och trycka in några url kommandon, vilka dessa url kommando är ser du i alla resource-filer, du listar ut dem.
* POST exempel för creating workItem : “name”: “shopping list”,

“description”: “bread, egg, cheese”

* Create User POST: **firstName**, **lastName**, **userName**
* Create Team POST: **name**, boolean **active**
* Create WorkItem POST: **name**, **description**
* Om allt har gått bra så ska det finnas data i din “mySQL”
* GET exempel för att söka user med hjälp av userNumber: users/1001, users/1002 och så vidare.

GOOD LUCK!

**README.md kopia**

Exekveringskrav:

\*IntelliJ IDEA 2017.3.5 (Community Edition)

\*java version "1.8.0\_131"

Java(TM) SE Runtime Environment (build 1.8.0\_131-b11) eller nyare

\*Database

MySQL Workbench 6.3

\*Postman

Version 6.0.10 (6.0.10)

Installationsguide:

Databasen:

Öppna MySQL Workbench 6.3 och skapa en ny serverconnection.

Öppna medföljd SQLscript.sql och kör

IntelliJ:

Öppna zip-filen i IntelliJ

Gå in i resourses mappen och sedan öppna application.properties och ändra ddl-auto,url,username och password till att anpassa er egna databas

Kör programmet från ProjektarbeteApplication för att starta Spring

Postman:

Starta Postman och ange http://localhost:8080 i url-fältet

För att lägga till användare till databasen ändra till POST och utöka url:en till http://localhost:8080/users

Postman Lathund:  
För att lägga till data i databasen använd POST funktionen i Postman och använd följande mallar.

Lägg till user

http://localhost:8080/users

{

"firstName": "firstname",

"lastName": "lastname",

"userName": "usernamelongerthan10characters"

}

Lägg till Team

http://localhost:8080/teams

{

"name": "teamname",

"active": true/false

}

Skapa WorkItem

http://localhost:8080/workitems

{

"name": "workitem\_name",

"description": "workitem\_description"

}

Skapa Issue för ett WorkItem

http://localhost:8080/workitems/6\*/issues

\*siffran är ett workitems id.

{

"description": "issue\_description"

}