City University London

IN3007 Individual Project
Final Year Project Report

Academic Year: 2014-15

Appendix A
Project Definition Document

by

Sergiu Tripon

Project supervisor: Dr. Vladimir Stankovic

Course

Computer Science

Project Title

An application for student access to university-related resources

Student Details

Name: Sergiu Tripon
E-mail: Sergiu.Tripon.1@city.ac.uk

Supervisor Details

Name: Vladimir Stankovic
E-mail: v.stankovic@city.ac.uk

Project Proposition by

Sergiu Tripon (the student)

Project Description

A fully responsive web application which allows university students to access university-related resources.

Arrangements for proprietary interests

No arrangements have been made.

Project acceptance promises

There are no particular promises made for the acceptance of the project.

Student Signature

Problem to be solved

My project will solve the lack of a responsive web application that is designed and developed for students enabling them to access university-related resources. This kind of "one stop" application doesn't exist at the moment, and it would be a benefit for the students to have it available. It will be a prototype using a small amount of real data due to data confidentiality issues.

The application will be responsive because "[w]e spend more time interacting with devices than with people" (Levin 2014, p. 1) and "[w]e often interact with more than one device at a time." (Levin 2014, p. 1)

Project Beneficiaries

The beneficiaries will be students, lecturers, library staff and events staff. Potential future developers (of the same or similar application) may also benefit due to the application being a prototype.

Project Objectives

This project shall provide a web application to be used by individuals within a university to access a wide range of university-related resources.

This project shall have the following sub-objectives:

Critical Priority

- The application will be fully responsive, usable on desktop/laptop, tablet or mobile phone.
- The application will provide the following functionality: pay course fees, check timetable and exam timetable, check exam results, reserve a book, book and pay for events, provide lecture feedback, check university map, message other users etc.

High Priority

- The application will undergo an analysis and design phase based on UML.
- The application will undergo an implementation/testing phase. This will consist of UML diagrams of the system architecture and acceptance/unit/exploratory test cases.

Moderate priority

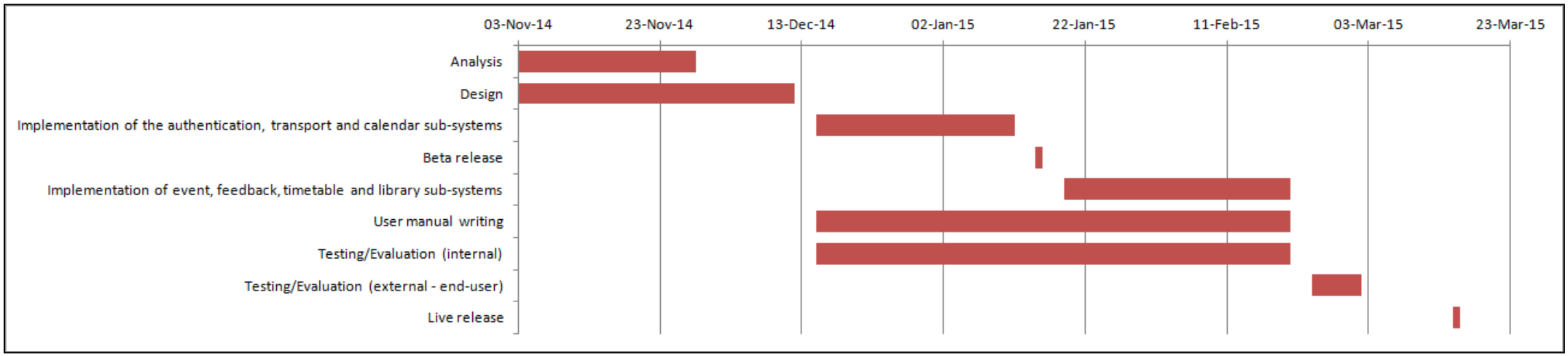
- The application will be cross-browser compatible.
- The application will undergo an evaluation phase by a end-user within the target audience.
- The application will have a user-manual.

To test whether the sub-objectives have been met or not, I will present intermediate versions of the outputs to my supervisor during our meetings.

Work Plan

The technology used throughout the project is PHP as the primary programming language, and MySQL as the back-end RDBMS. HTML5, CSS, JavaScript, jQuery and the Bootstrap front-end framework will also be used.

ID	Work Packages	Start date	Duration	Due date
1	Analysis	03-Nov-14	25	28-Nov-14
2	Design	03-Nov-14	39	12-Dec-14
3	Implementation of the authentication, transport and calendar sub-systems	15-Dec-14	28	12-Jan-15
4	**Beta release**	**15-Jan-15**	**1**	**16-Jan-15**
5	Implementation of event, feedback, timetable and library sub-systems	19-Jan-15	32	20-Feb-15
6	User manual writing	15-Dec-14	67	20-Feb-15
7	Testing/Evaluation (internal)	15-Dec-14	67	20-Feb-15
8	Testing/Evaluation (external - end-user)	23-Feb-15	7	02-Mar-15
9	**Live release**	**15-Mar-15**	**1**	**16-Mar-15**



Work Packages

Analysis and Design

Outputs: identify the goals of the project, and figure out procedures to achieve them efficiently, definition of the architecture, interfaces, and data for the project to satisfy specified requirements

Implementation of the authentication, transport and calendar sub-systems

Outputs: various functionality

Implementation of event, feedback, timetable and library sub-systems

Outputs: various functionality

User manual

Outputs: manual with instructions on how to perform advanced tasks and FAQs

Testing/Evaluation (internal)

Outputs: results from acceptance/unit/exploratory test cases

Testing/Evaluation (external - end-user)

Outputs: results and feedback from external end-user

Resources to be used: supervisor, software, books, time, money

Project Risks

Technical

One of the programming languages that I will use throughout the project is PHP, a language which I don't have a lot of experience in at the moment. There is the possibility of running into a PHP-related issue, which could take me longer than usual to solve.

To prevent this technical risk, as soon as I've agreed a project with my supervisor, I started learning PHP and practicing on the basis of the future application.

Complexity

During the production phase, I will be using different programming languages for different purposes. They are necessary due to the complexity of the project and the user interface I am aiming to achieve, but this can make the project quite complex, which could be a negative. The responsive aspect of the application could also be a risk as Michal Levin states "Keep adhering to the basic principles of user-centred design, and beware of getting too device-enchanted, which can pull you into the tempting yet risky technology-driven approach—especially with the pace at which connected technology is developing and new, exciting devices are proliferating, offering people new means to reach a goal." (Levin 2014, p. 138)

Since August, I have been thinking, researching and analysing whether I can achieve the project with its complexity level in mind, and having analysed each part of the project, I came to the conclusion that I will be able to cope with the complexity and achieve everything as planned.

Resources

Resources are necessary for any project. To secure a successful project, I need various resources like the help of my supervisor, books, software, time or money. If one of the resources is missing, the potential of the project can be affected, and therefore it can become a serious risk.

One way to be sure that I won't be lacking any resources, is securing these resources in advance which I already did. I made sure that I chose a supervisor that above all is responsive with feedback and help, I researched and looked for books that will help as I progress, and I made sure that I have the software tools necessary for the project.

Requirements

A well thought out requirements gathering is essential, but sometimes unexpected things can occur like requirements changing at important stages of the project or misunderstandings with the user. Requirements risks are very important to avoid, due to the gravity of the impact they can have on the project.

My approach to avoid a potential requirements risk is communication. Once I gather the requirements, the communication between myself and the user won't stop, as that will enable me to seek feedback and opinions as the development work progresses, through agile methodology.

Initial assumptions

Initial assumptions have to be made regardless, but they can be a risk. Before, now, and in the future, I make assumptions. I assume what will be difficult and what will be easy, what will require a lot of time and what will be done in a matter of hours. Mistakes are made when assumptions are the basis for planning, because if the assumptions are wrong, the plan is affected.

To avoid my plan being affected by wrong assumptions, it won't totally rely on them. Yes, I will make assumptions, but while planning I will set aside extra time to amend any wrong assumptions I may have made.

Bibliography

LEVIN, M. (2014) *Designing Multi-Device Experiences.* 1st Ed. Sebastopol: O'Reilly Media.

Research Ethics Checklist
Department of Computer Science BSc, MSc, MA Projects

If the answer to any of the following questions (1 – 3) is NO, your project needs to be modified.		*Delete as appropriate*
1.	Does your project pose only minimal and predictable risk to you (the student)?	**Yes**
2.	Does your project pose only minimal and predictable risk to other people affected by or participating in the project?	**Yes**
3.	Is your project supervised by a member of academic staff of the School of Informatics or another individual approved by the module leaders?	**Yes**

If the answer to either of the following questions (4 – 5) is YES, you MUST apply to the University Research Ethics Committee for approval. (You should seek advice about this from your project supervisor at an early stage.)		*Delete as appropriate*
4.	Does your project involve animals?	**No**
5.	Does your project involve pregnant women or women in labour?	**No**

If the answer to the following question (6) is YES, you MUST complete the remainder of this form (7 – 19). If the answer is NO, you are finished.		*Delete as appropriate*
6.	Does your project involve human participants? For example, as interviewees, respondents to a questionnaire or participants in evaluation or testing?	**Yes**

If the answer to any of the following questions (7 – 13) is YES, you MUST apply to the Informatics Research Ethics Panel for approval and your application may be referred to the University Research Ethics Committee. (You should seek advice about this from your project supervisor at an early stage.)		*Delete as appropriate*
7.	Could your project uncover illegal activities?	**No**
8.	Could your project cause stress or anxiety in the participants?	**No**
9.	Will you be asking questions of a sensitive nature?	**No**
10.	Does your project rely on covert observation of the participants?	**No**
11.	Does your project involve participants who are under the age of 18?	**No**
12.	Does your project involve adults who are vulnerable because of their social, psychological or medical circumstances (vulnerable adults)?	**No**
13.	Does your project involve participants who have learning difficulties?	**No**

The following questions (14 – 16) must be answered YES, i.e. you MUST COMMIT to satisfy these conditions and have an appropriate plan to ensure they are satisfied.	*Delete as appropriate*
14. Will you ensure that participants taking part in your project are fully informed about the purpose of the research?	**Yes**
15. Will you ensure that participants taking part in your project are fully informed about the procedures affecting them or affecting any information collected about them, including information about how the data will be used, to whom it will be disclosed, and how long it will be kept?	**Yes**
16. When people agree to participate in your project, will it be made clear to them that they may withdraw (i.e. not participate) at any time without any penalty?	**Yes**

The following questions (17 – 19) must be answered and the requested information provided.	*Delete as appropriate*
17. Will consent be obtained from the participants in your project? Consent from participants will be necessary if you plan to gather personal, medical or other sensitive data about them. "Personal data" means data relating to an identifiable living person; e.g. data you collect using questionnaires, observations, interviews, computer logs. The person might be identifiable if you record their name, username, student id, DNA, fingerprint, etc. *If **YES**, provide the consent request form that you will use and indicate who will obtain the consent, how are you intending to arrange for a copy of the signed consent form for the participants, when will they receive it and how long the participants will have between receiving information about the study and giving consent, and when the filled consent request forms will be available for inspection (**NOTE**: subsequent failure to provide the filled consent request forms will automatically result in withdrawal of any earlier ethical approval of your project):* I will contact the participant and meet up with them in order to provide them with a copy of the information sheet and consent form. They will have 1 week to read the information sheet, and if they agree, sign the consent form. The filled information sheet and consent request forms will be available for inspection from 27th October, the date when the Project Definition Document is due for submission.	**Yes**
18. Have you made arrangements to ensure that material and/or private information obtained from or about the participating individuals will remain confidential? *Provide details:* No private information will kept once the evaluation/testing is done by the individual. Their account will be deleted, and the data used will be erased. I will not talk about any of the private information in my Project Report document; it will be purely about the feedback and test results that the individual provided in regards to the application.	**Yes**
19. Will the research be conducted in the participant's home or other non-University location?	**Yes**

*If **YES**, provide details of how your safety will be preserved:*

I have been in a friendship with the participant for 2 years. I feel safe in their company, and have no doubt whatsoever that my safety will definitely preserved.

EVALUATION/TESTING INFORMATION SHEET

Headed paper – clear identification of the University as the responsible institution

City University London

Title of Study:

Evaluation/test of: An application for student access to university-related resources

I would like to invite you to take part in the evaluation/testing phase of a web application developed for students. Before you decide whether you would like to take part it is important that you understand why the evaluation/testing phase is being done and what it would involve for you. Please take time to read the following information carefully and discuss it with others if you wish. Ask me if there is anything that is not clear or if you would like more information.

What is the purpose of the study?
The purpose of this study is to evaluate/test a web application that I will be developing for my dissertation. The application is a Student Portal that provides students with access to university-related resources. The application will be fully responsive, so it would be ideal if you could evaluate/test it with not just one device, but a number of devices e.g. laptop/desktop, tablet, mobile phone.

Please note that the application at the time of the evaluation/testing phase will still be a prototype, which may contain bugs or lack certain functionality. Provide any feedback on any aspect of the application that you may have.

Why have I been invited?
You have been invited because firstly, you are a university student which is the target audience of the application, and secondly because you are a university student that does not study at City University London as that provides a different perspective of how the structure of another university may be different and whether the application would be suitable to any university, not just City University London as it will be built to be. It will also give a clear indication on which features may not suit every university, and whether there are any university-specific features that may have to be developed in a bespoke manner for a specific university in order for the application to be used successfully.

Do I have to take part?
Participation in the project is voluntary, and you can choose not to participate in part of or the entire project. You can withdraw at any stage of the project without being penalised or disadvantaged in any way.

It is up to you to decide whether or not to take part. If you do decide to take part you will be asked to sign a consent form. If you decide to take part you are still free to withdraw at any time and without giving a reason.

What will happen if I take part?

- *How long will the participant be involved:* 1 day, maximum 2 days
- *How long will the research study last:* 1 day, maximum 2 days
- *How often will the participants meet the researcher/s:* Once or twice
- *How long will the meetings with the researcher/s be:* 1-5 hours
- *What exactly will happen*: The user will load the application on a web browser. The user will be testing/evaluating each feature that the application has against a set of test cases. The user will have access to the application's user manual at any time. The user will provide feedback on the application as well as stating on whether they would use such an application, and if it suits the target audience.
- *What is the research method used:* There won't be any research method, but the participant will be provided with test cases on paper or electronically stating exactly what they need to test. There will also be notes taken.

- *Where is the research taking place:* The participant's home

Expenses and Payments (if applicable)

- There will be no travel expenses or rewards

What do I have to do?

The participant will be testing/evaluating the application. They will navigate through the application and test it against the test cases provided. They will also comment and give their opinion on what they think about it, whether it suits the target audience and if they would use it.

What are the possible disadvantages and risks of taking part?

There will be no disadvantages or risks.

What are the possible benefits of taking part?

The possible benefits will be the participant becoming a user of the application.

What will happen when the research study stops?

Any private information will be totally erased and the participant will have to be present when this happens.

Will my taking part in the study be kept confidential?

- *Who will have access to the information:* I and City University London will have access to the anonymous information. This information will solely be the results of test cases and any feedback that I will receive, but they will be kept completely anonymous. The information will be used towards the Project Report Document. I will be the only one that will have access to the personal information of the participant which will be erased immediately after the evaluation/testing phase has been completed and while the participant is present.
- *Audio/video recording/photographs*: There will be no Audio/video recording/photographs
- *Future use of personal information:* There will be no Future use of personal information
- *Data archiving/sharing:* There will be no Data archiving/sharing
- *Any restrictions on confidentiality:* There will be no restrictions on confidentiality
- The personal data will be stored in a test MySQL database. At the end of the exercise, the database will be completely deleted together with its tables in the presence of the participant.

What will happen to results of the research study?

The results will be used towards my Project Report Document. Throughout the document, the identity of the participant will be kept completely anonymous. The participant will be able to obtain a copy of the report, if they want to. To obtain one, they will have to ask.

What will happen if I don't want to carry on with the study?

The participant is free to withdraw from the evaluation/testing phase without an explanation or penalty.

What if there is a problem?

If there is a problem, you can always speak to me and I will try to deal with it in the best possible way.

If you have any problems, concerns or questions about this study, you should ask to speak to a member of the research team. If you remain unhappy and wish to complain formally, you can do this through the University complaints procedure. To complain about the study, you need to phone 020 7040 3040. You can then ask to speak to the Secretary to Senate Research Ethics Committee and inform them that the name of the project is: **An application for student access to university-related resources**

You could also write to the Secretary at:

Removed as it's confidential.

City University London holds insurance policies which apply to this study. If you feel you have been harmed or injured by taking part in this study you may be eligible to claim compensation. This does not affect your legal rights to seek compensation. If you are harmed due to someone's negligence, then you may have grounds for legal action.

Who has reviewed the study?
This study has been approved by City University London Research Ethics Committee

Further information and contact details

Student Details

Name: Sergiu Tripon
E-mail: Sergiu.Tripon.1@city.ac.uk

Supervisor Details

Name: Vladimir Stankovic
E-mail: v.stankovic@city.ac.uk

Thank you for taking the time to read this information sheet.

EVALUATION/TESTING CONSENT FORM

Headed paper – clear identification of the University as the responsible institution

City University London

Title of Study: **Evaluation/test of:** An application for student access to university-related resources

Please initial box

1.	I agree to take part in the above City University London research project. I have had the project explained to me, and I have read the participant information sheet, which I may keep for my records. I understand this will involve the following: • Evaluating/Testing the application given against the test cases provided • Giving your opinion on whether the application suits the target audience, and whether you would use the application or not provided that it works for your university, hypothetically speaking	
2.	This information will be held and processed for the following purpose(s): • Potential changes to the application following the phase • Project Report Document The identity of the participant will not be used in any of the above purposes. I understand that any information I provide is confidential, and that no information that could lead to the identification of any individual will be disclosed in any reports on the project, or to any other party. No identifiable personal data will be published. The identifiable data will not be shared with any other organisation.	
3.	I understand that my participation is voluntary, that I can choose not to participate in part or all of the project, and that I can withdraw at any stage of the project without being penalized or disadvantaged in any way.	
4.	In rare occasions, City University London may contact you to verify your participation in the study. The information gathered will be kept confidential and will not be included, in any text to be freely circulated.	
5.	I agree to City University London recording and processing this information about me. I understand that this information will be used only for the purpose(s) set out in this statement and my consent is conditional on the University complying with its duties and obligations under the Data Protection Act 1998.	
6.	I agree to take part in the above study.	

Pratiksha Vekaria 25/10/2014

_____ _____ _____

Name of Participant Signature Date

When completed, 1 copy for participant; 1 copy for researcher file.

City University London

IN3007 Individual Project
Final Year Project Report

Academic Year: 2014-15

Appendix B
Requirements document

by

Sergiu Tripon

Project supervisor: Dr. Vladimir Stankovic

Table of Contents

1. Introduction

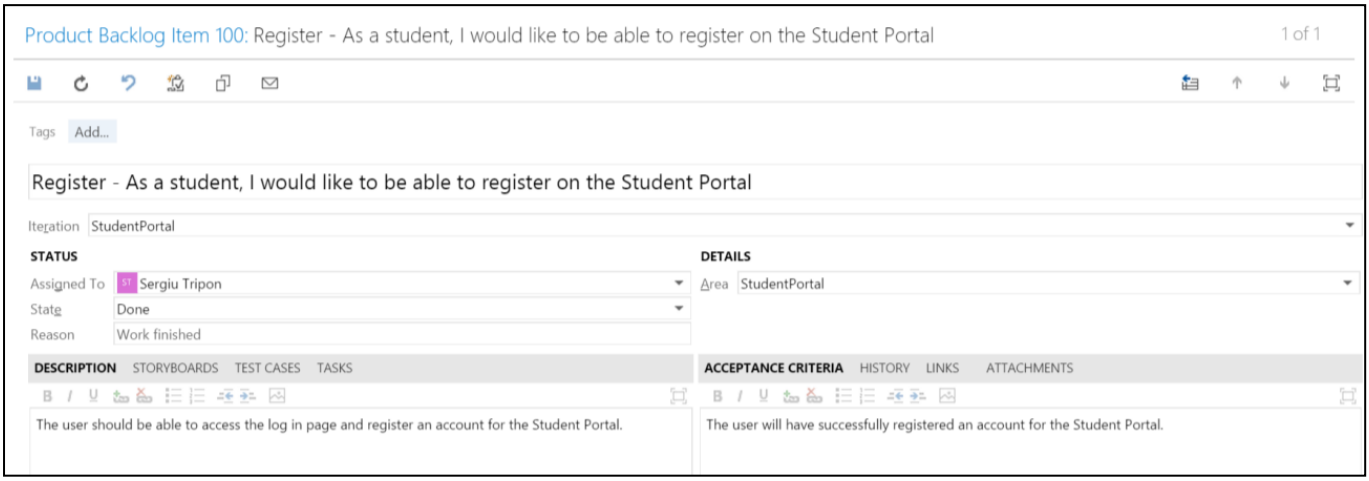
This document showcases each requirement defined for implementation within the Student Portal application. It consists of requirement specifications for each account type of the system: Student, Academic staff, Administrator. Most of the requirements are different by account type, but there are also some requirements which are common between the users of the system. These requirements are detailed after the individual account type specific requirements.

2. Requirements

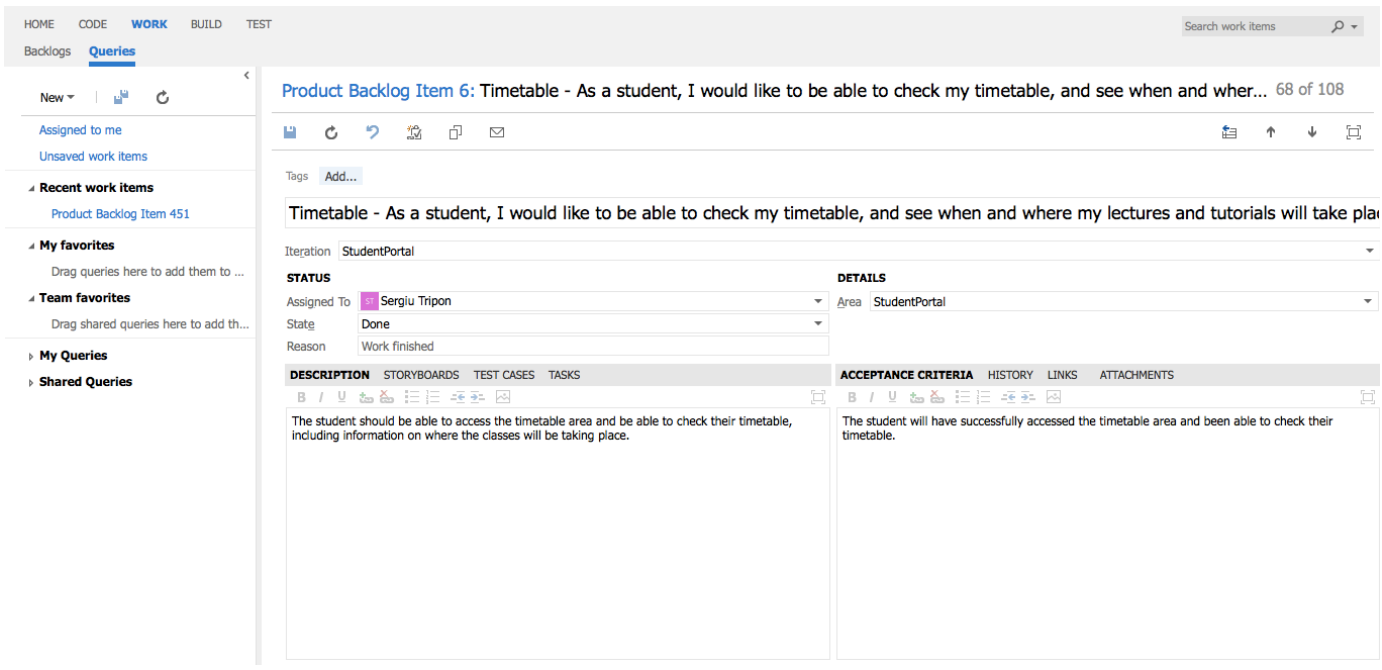
2.1. Account type specific requirements

2.1.1. Student

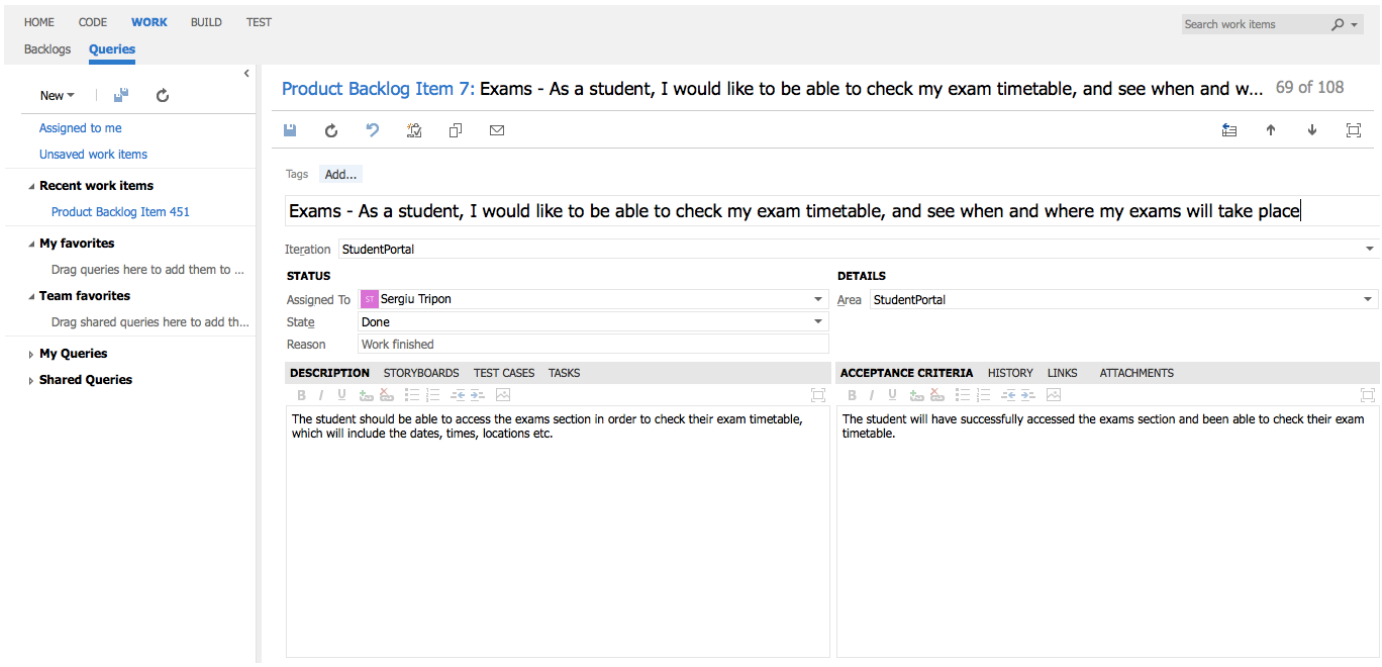
Register - detailed view of requirement



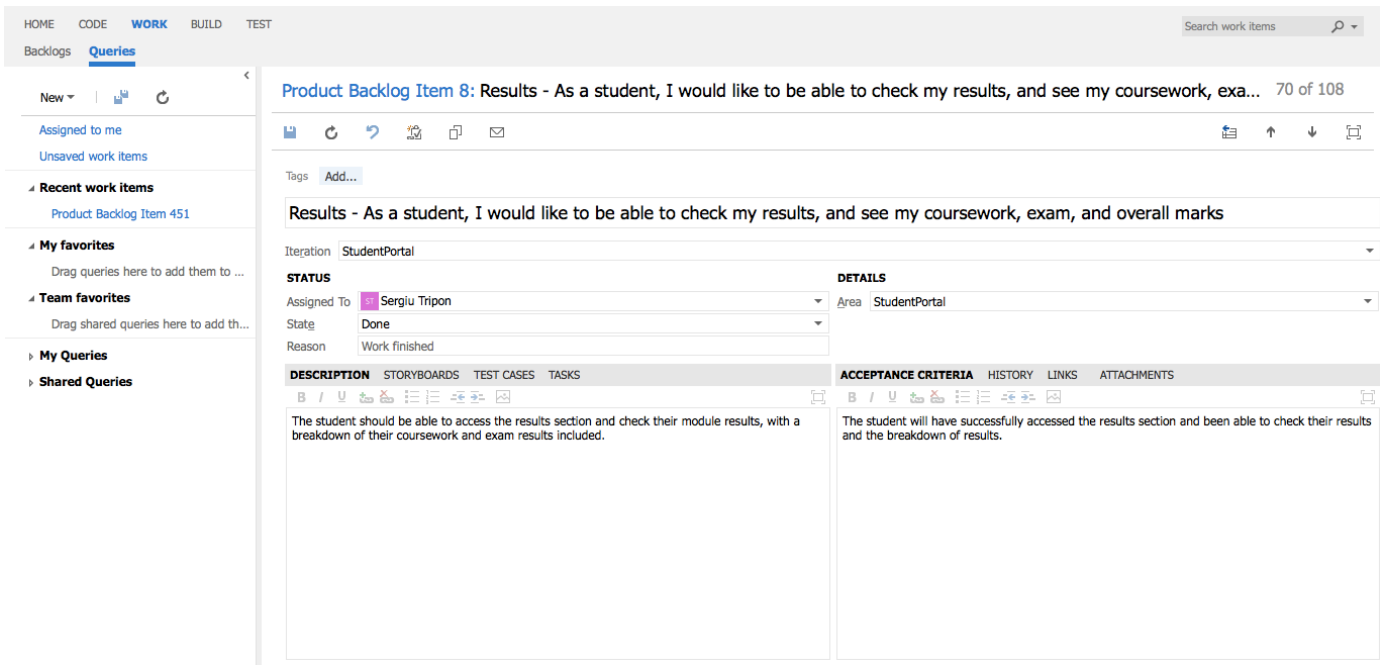
Timetable - detailed view of requirement



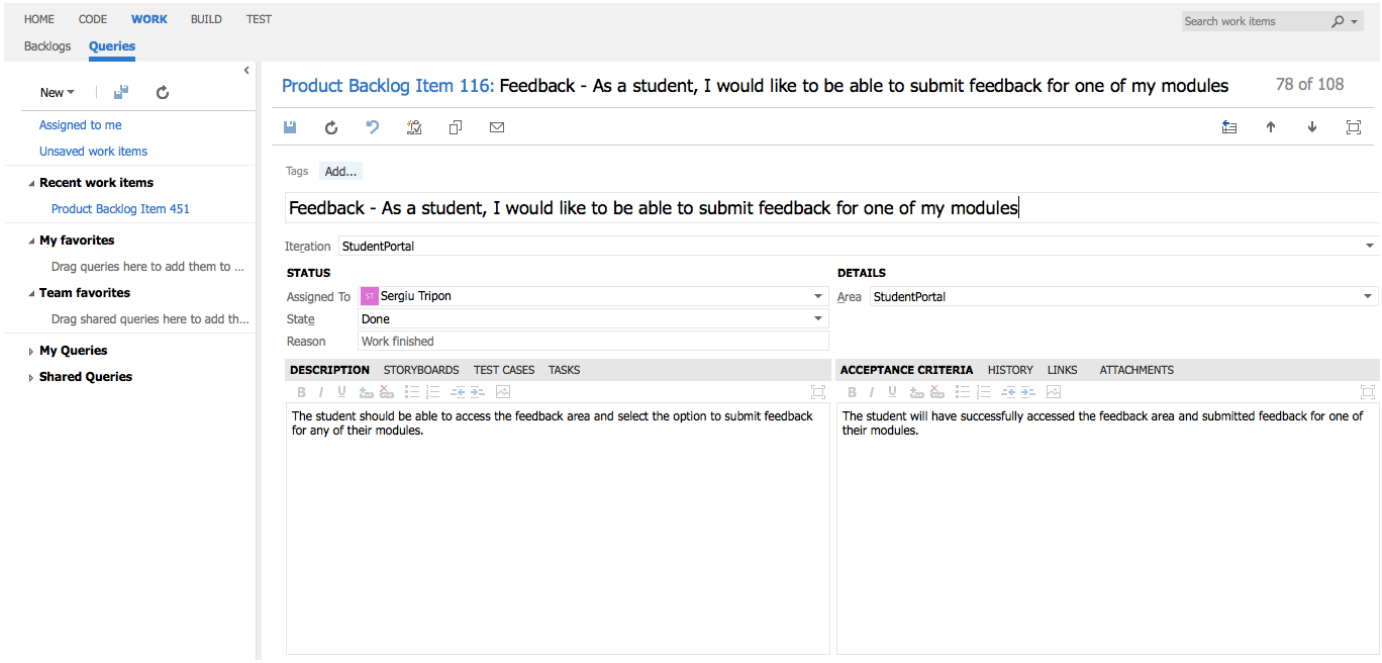
Exams - detailed view of requirement



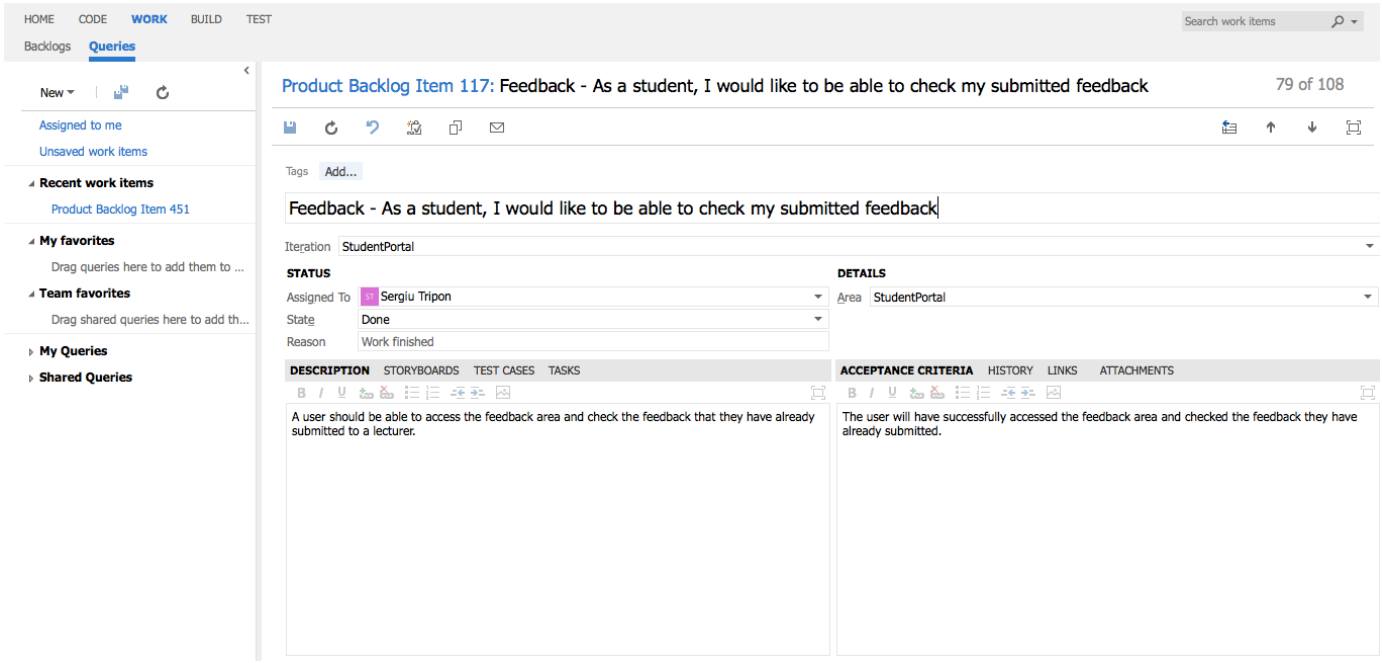
Results - detailed view of requirement



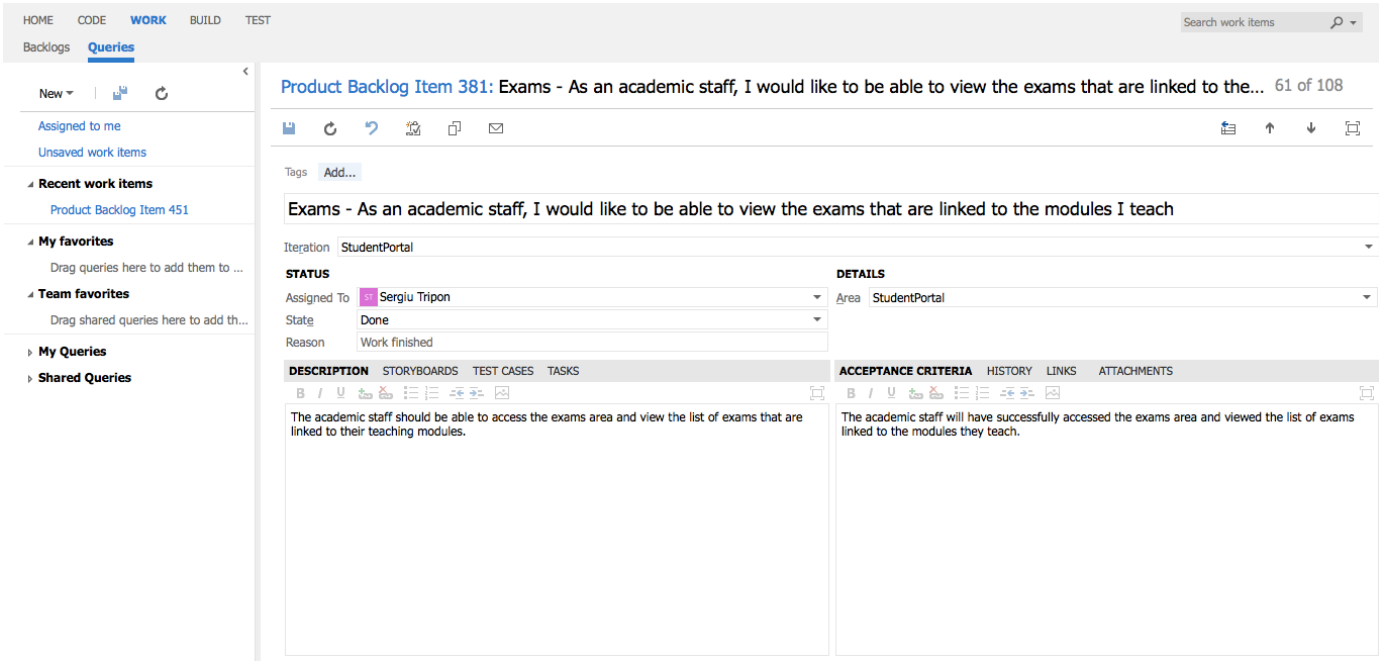
Feedback - detailed view of requirement



Feedback - detailed view of requirement

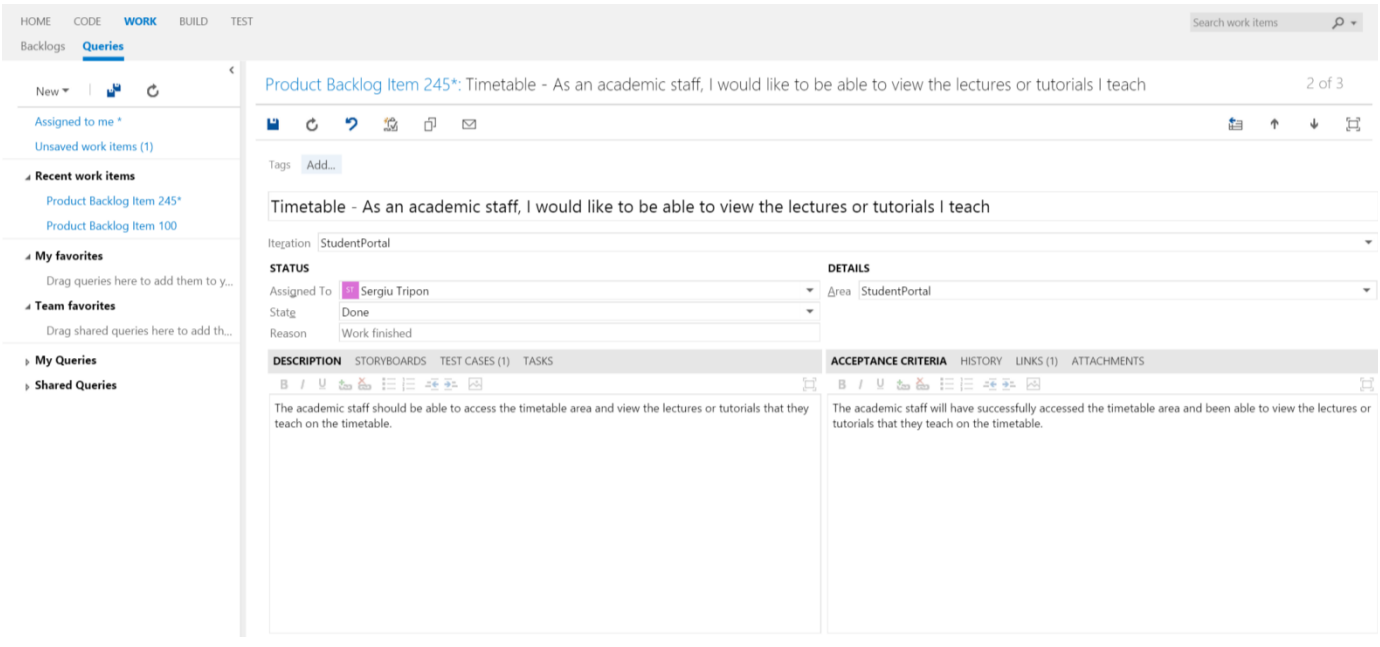


Exams - detailed view of requirement

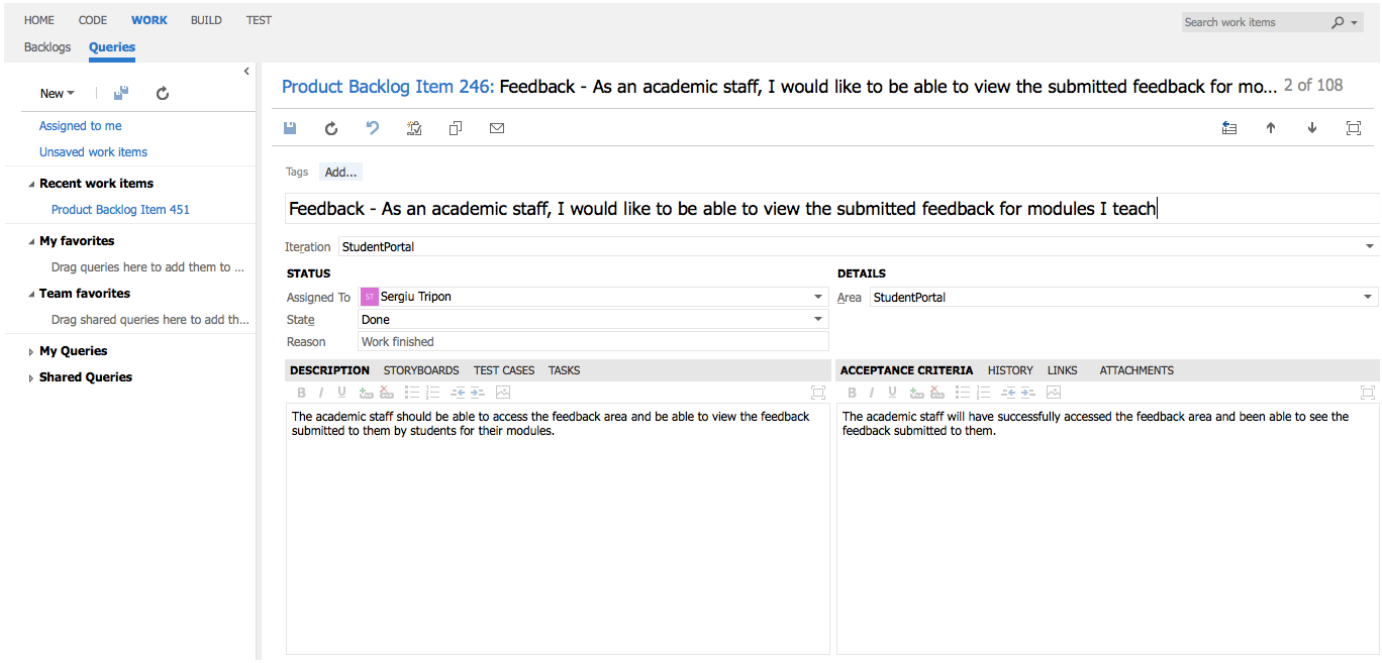


2.1.2. Academic staff

Timetable - detailed view of requirement

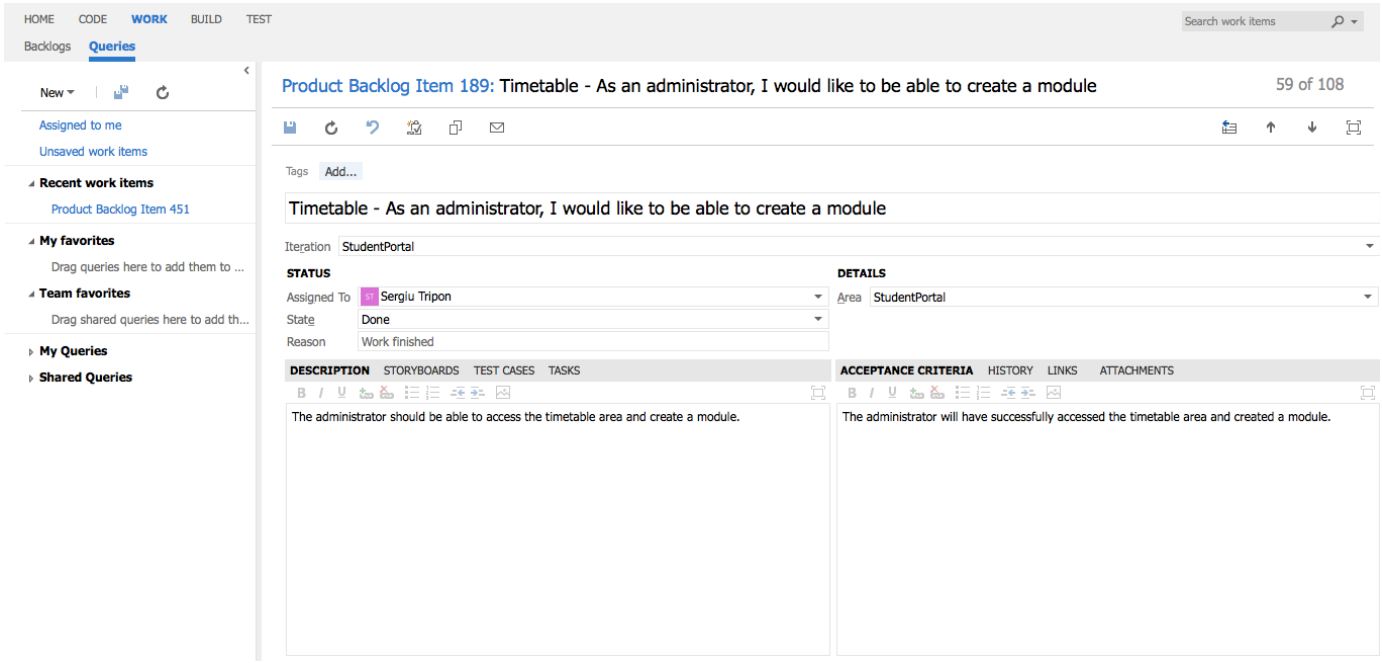


Feedback - detailed view of requirement

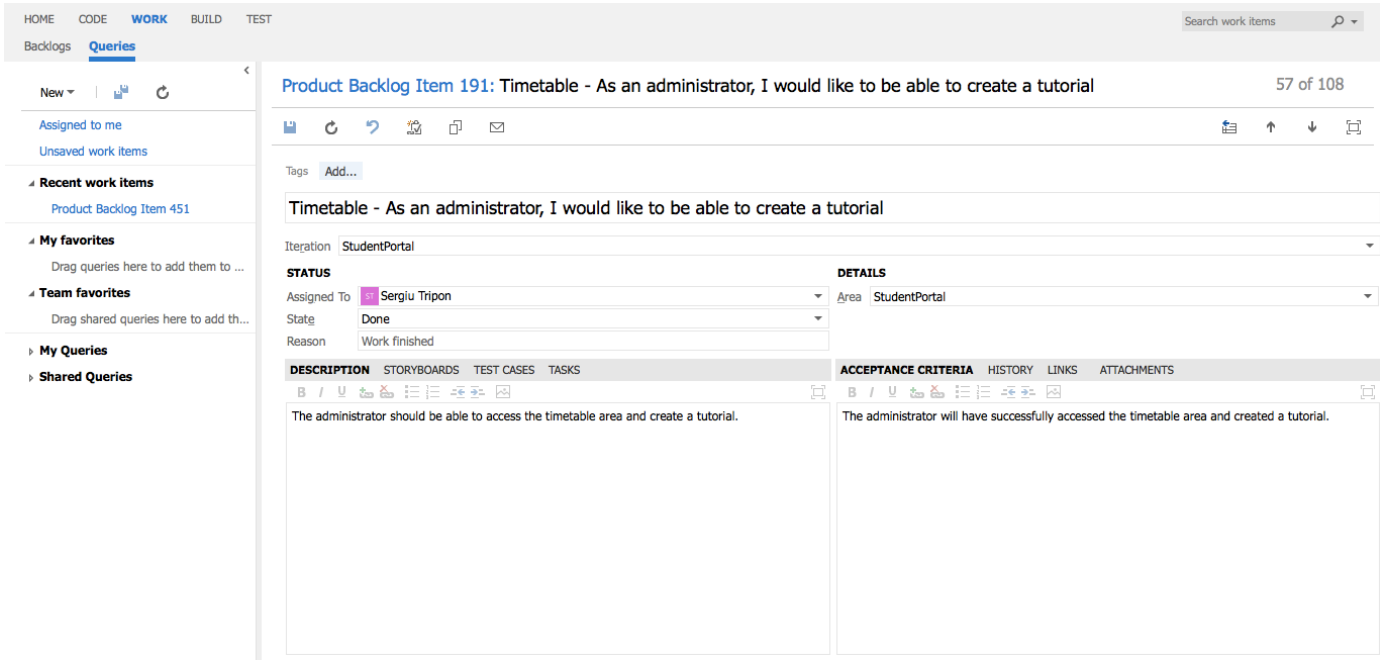


2.1.3. Administrator

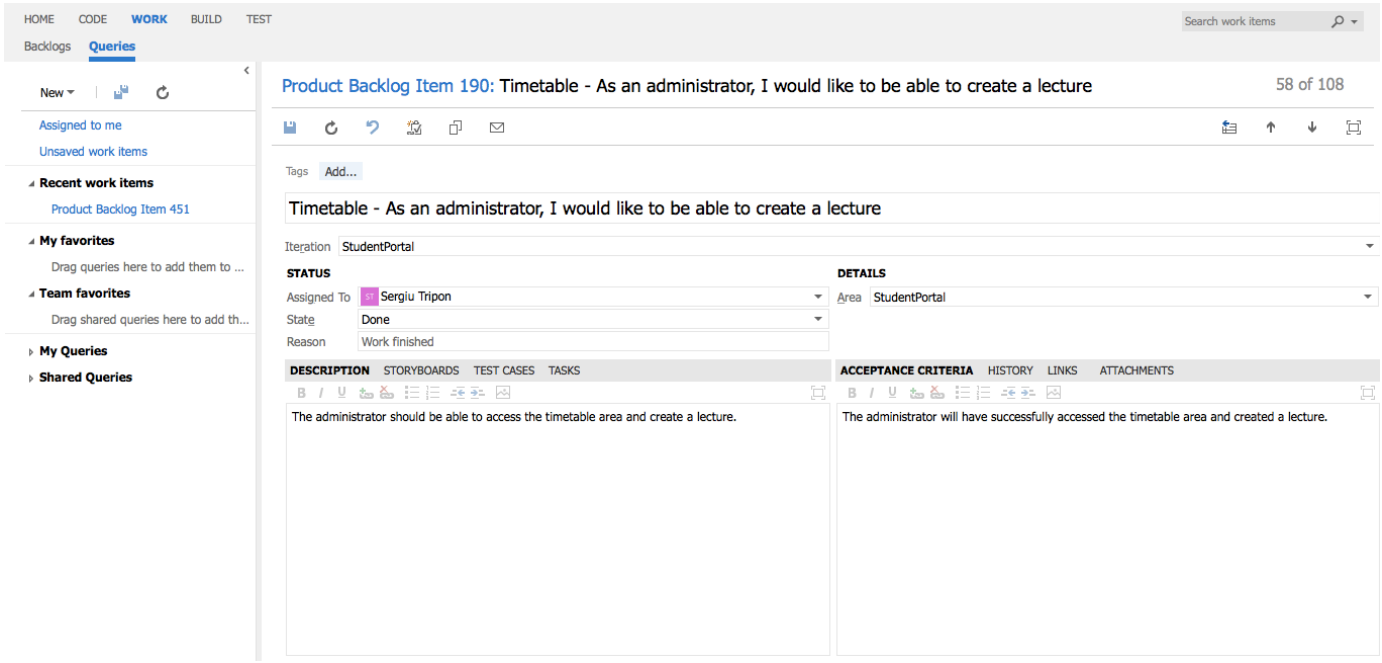
Timetable - detailed view of requirement



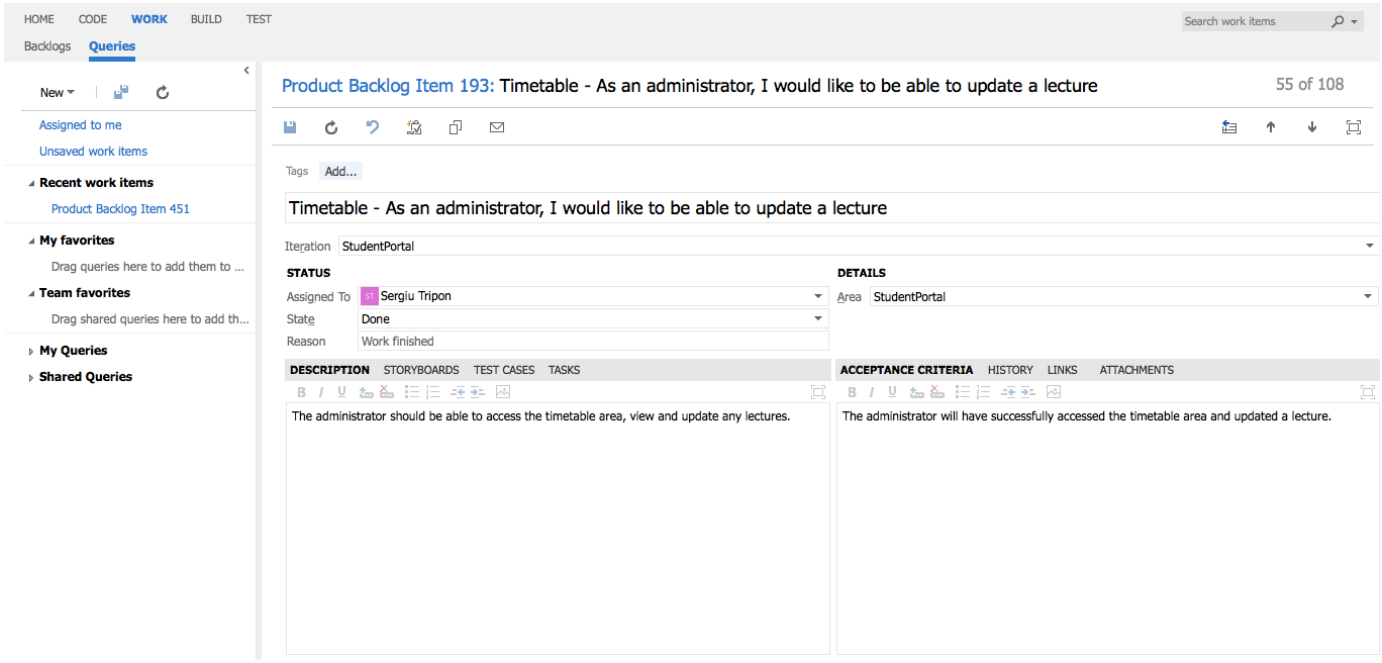
Timetable - detailed view of requirement



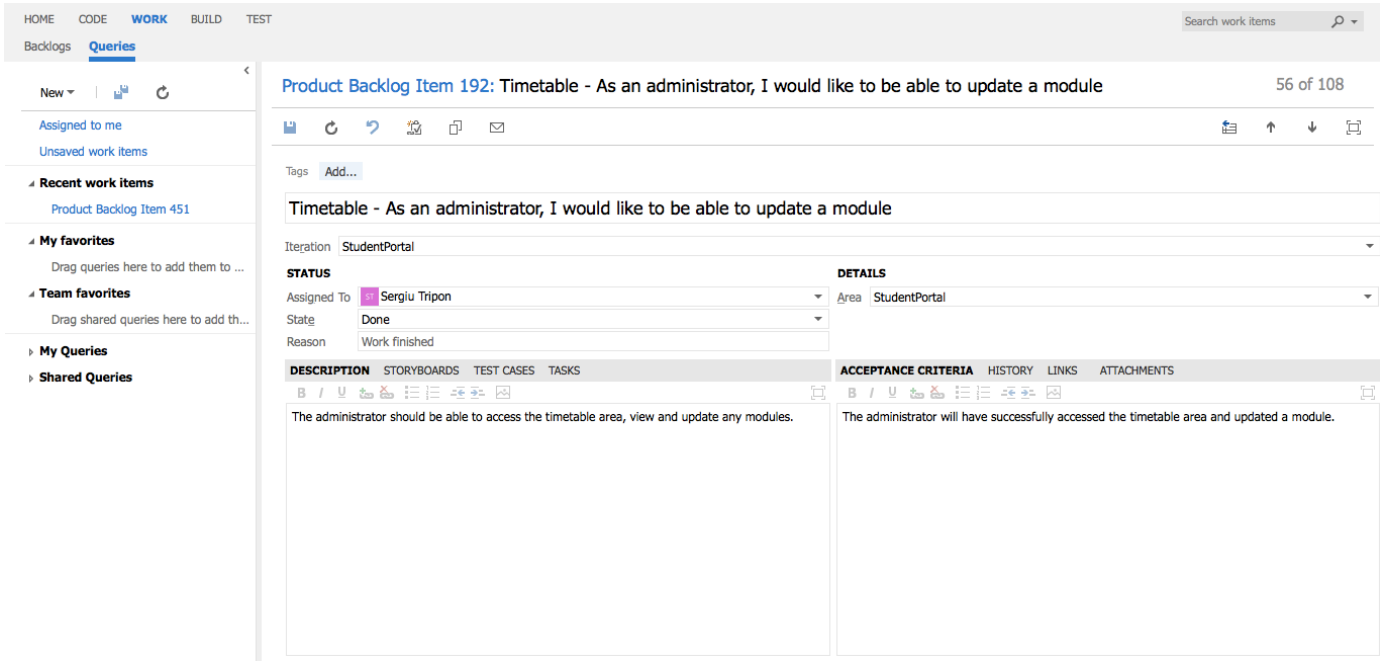
Timetable - detailed view of requirement



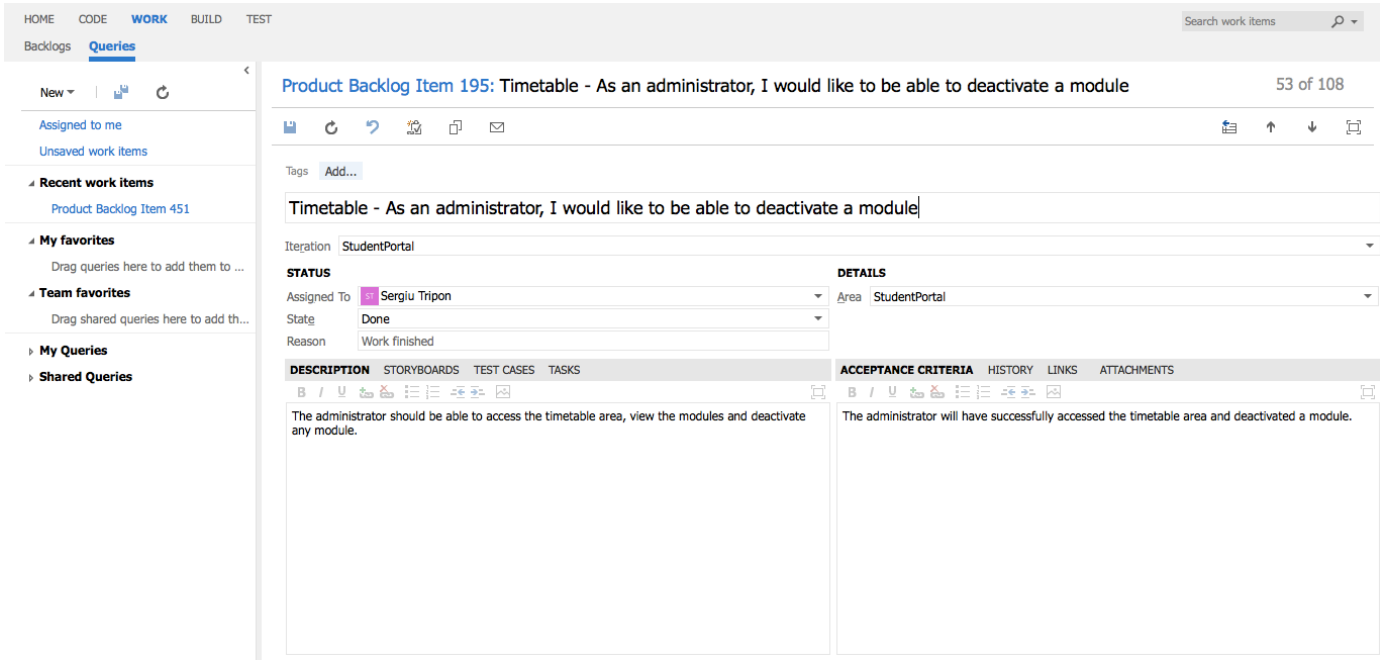
Timetable - detailed view of requirement



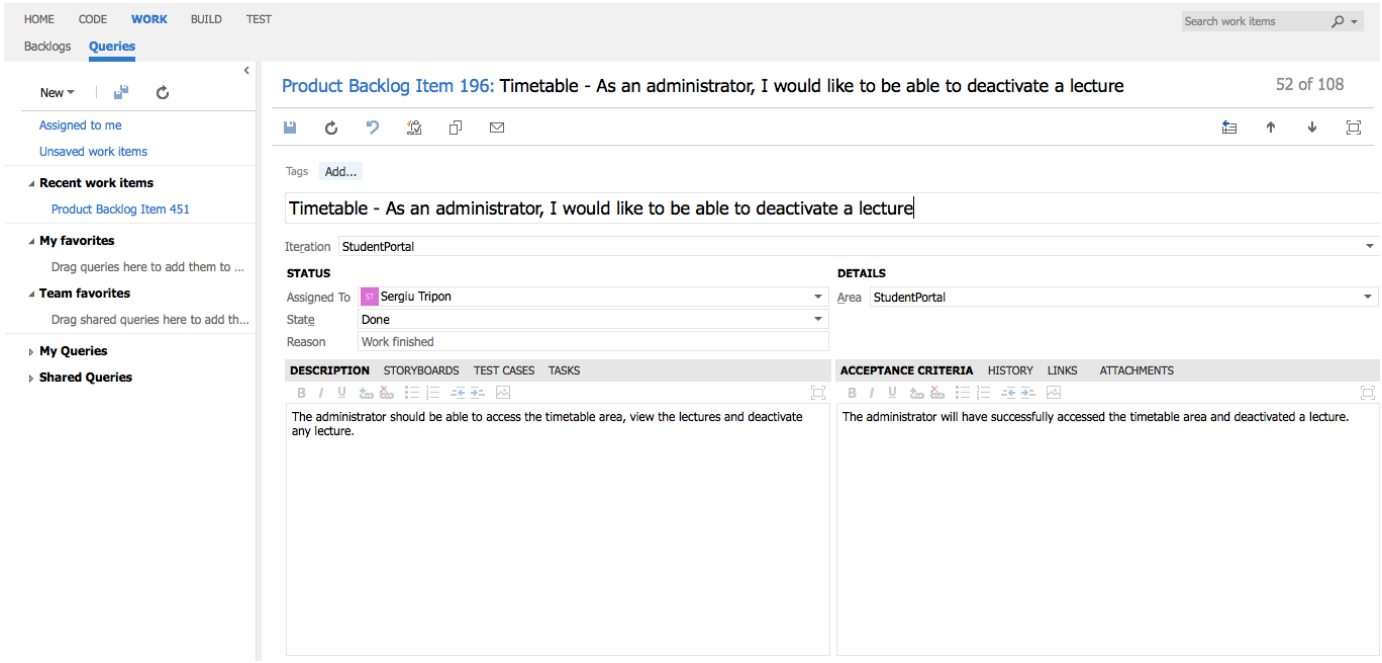
Timetable - detailed view of requirement



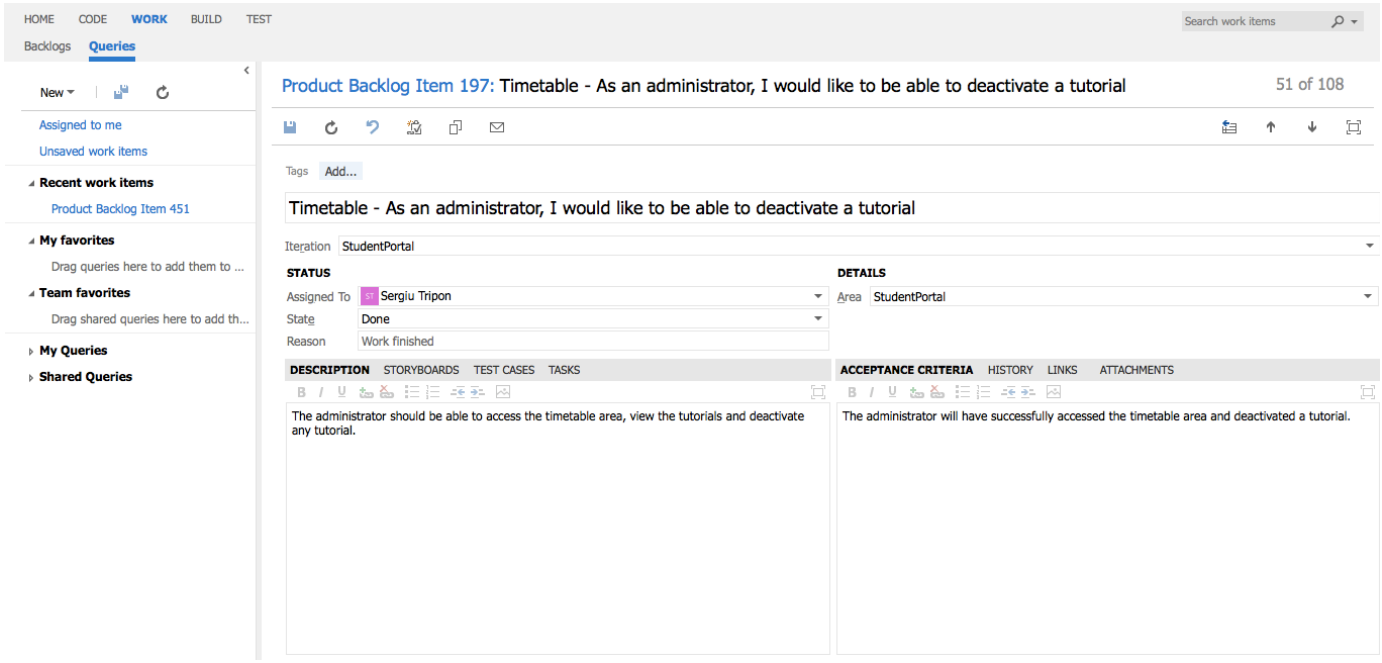
Timetable - detailed view of requirement



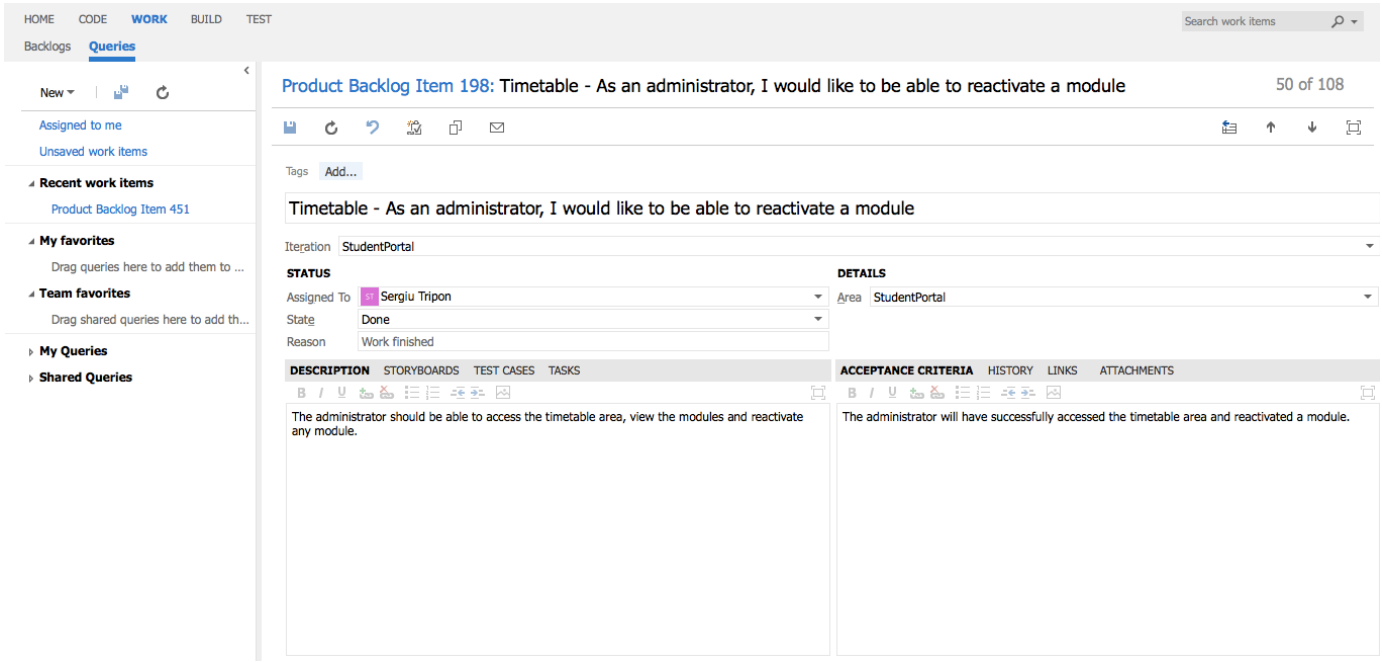
Timetable - detailed view of requirement



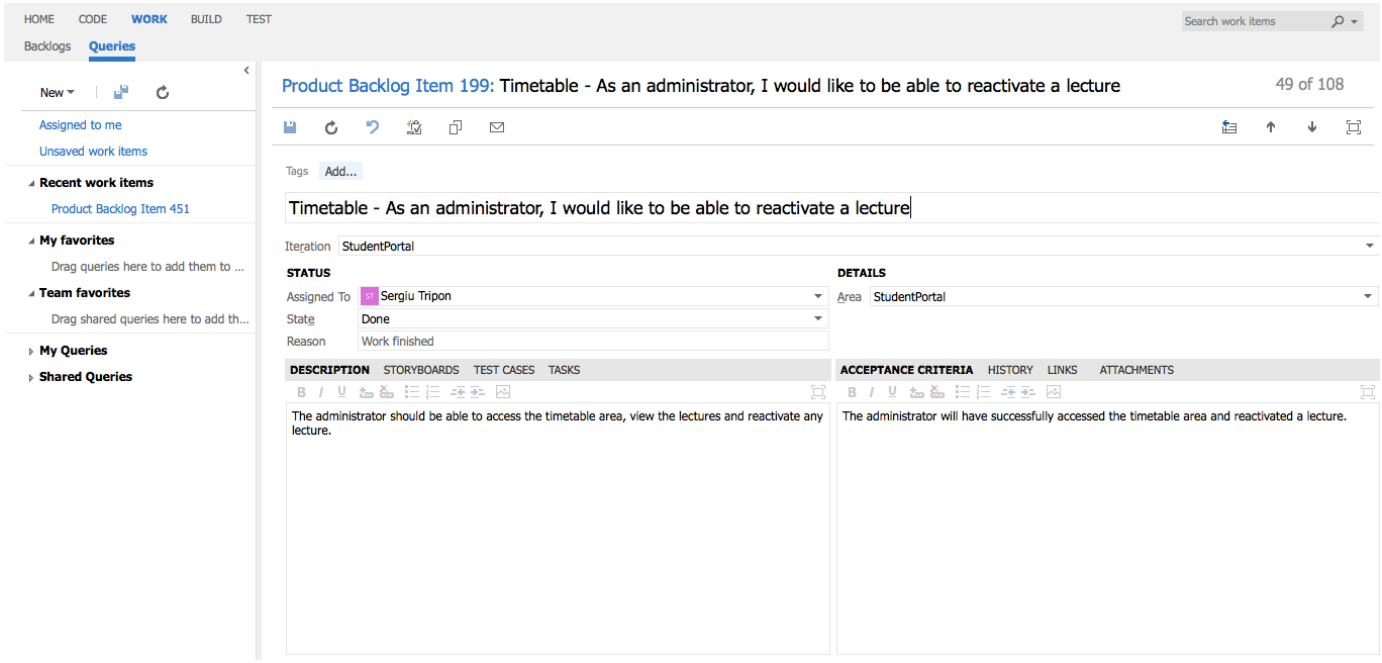
Timetable - detailed view of requirement



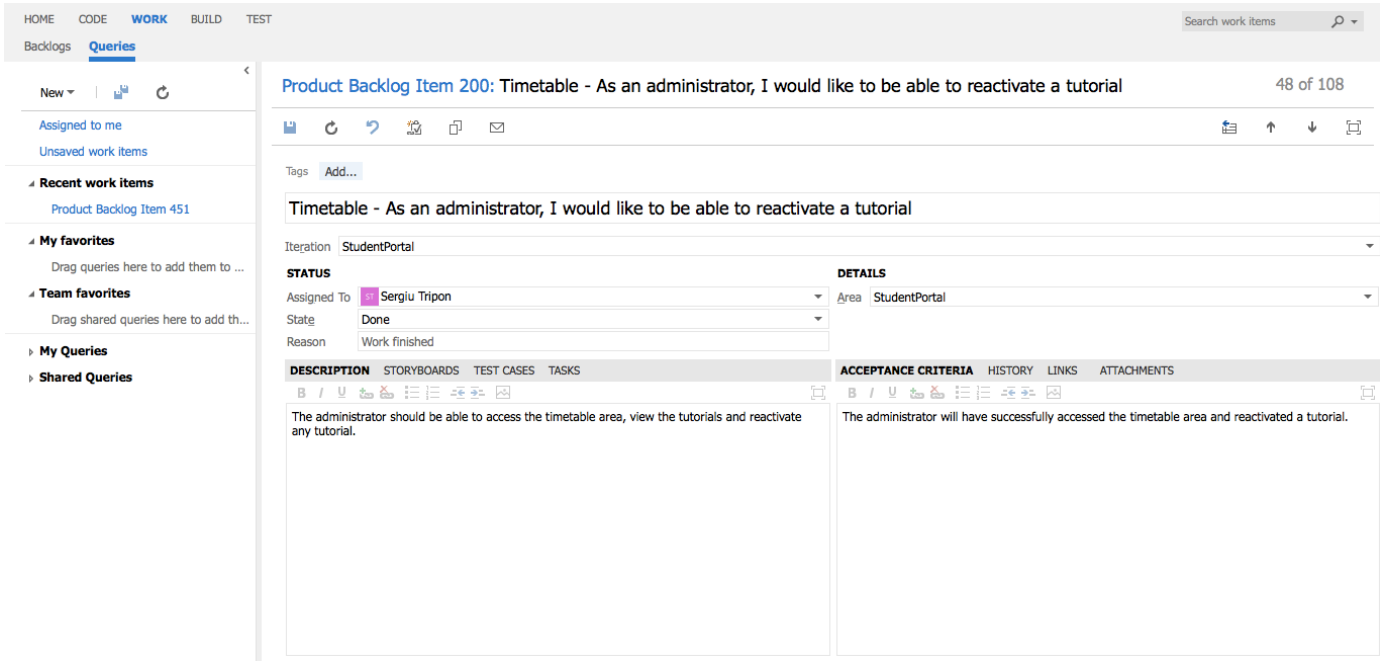
Timetable - detailed view of requirement



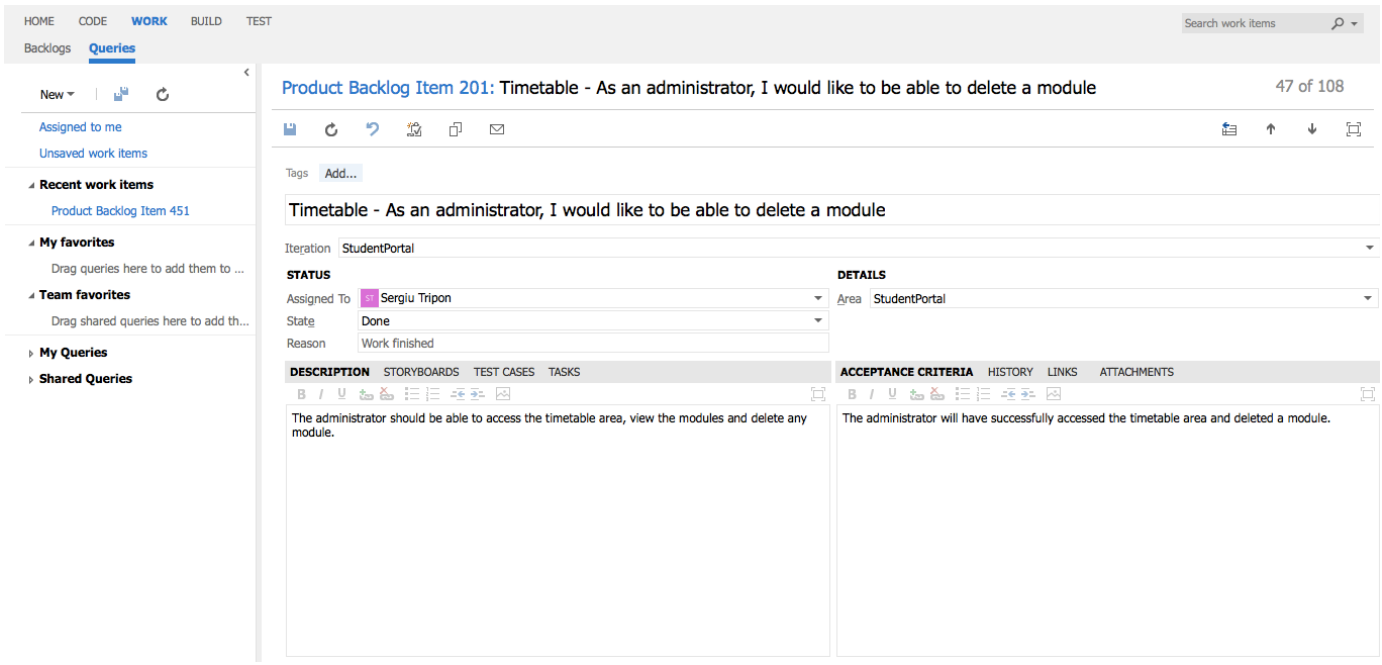
Timetable - detailed view of requirement



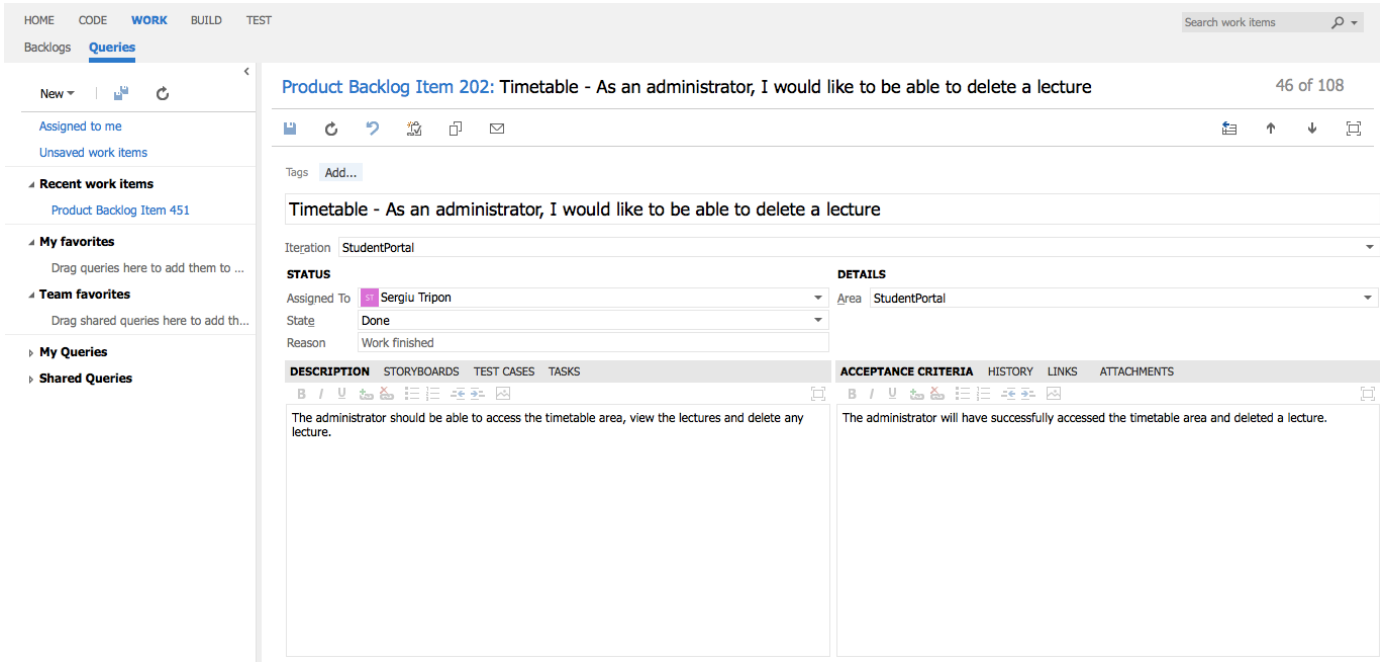
Timetable - detailed view of requirement



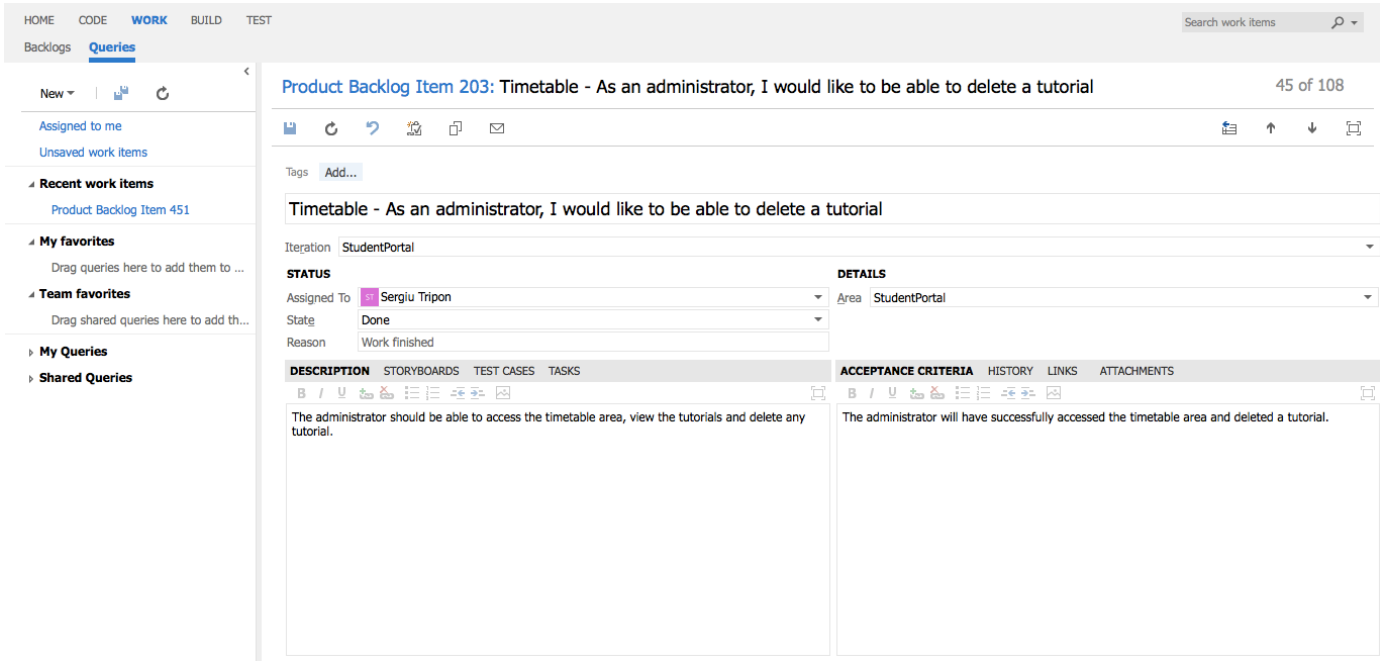
Timetable - detailed view of requirement



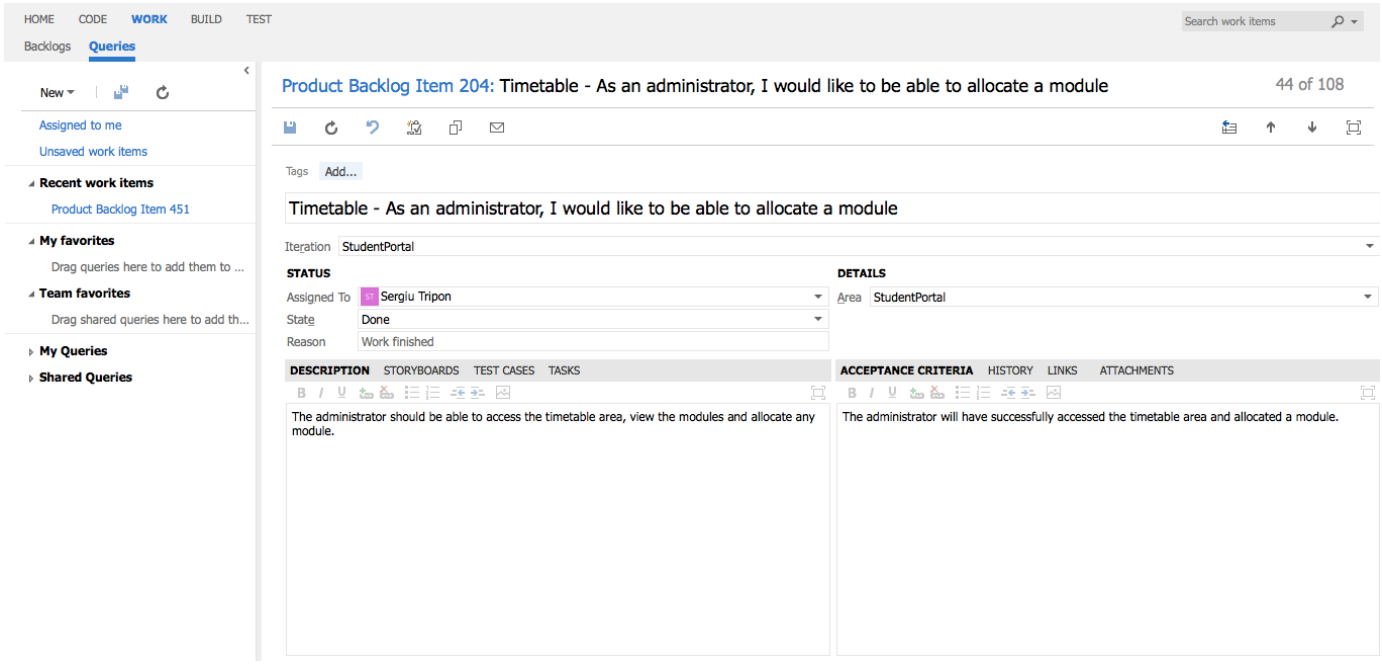
Timetable - detailed view of requirement



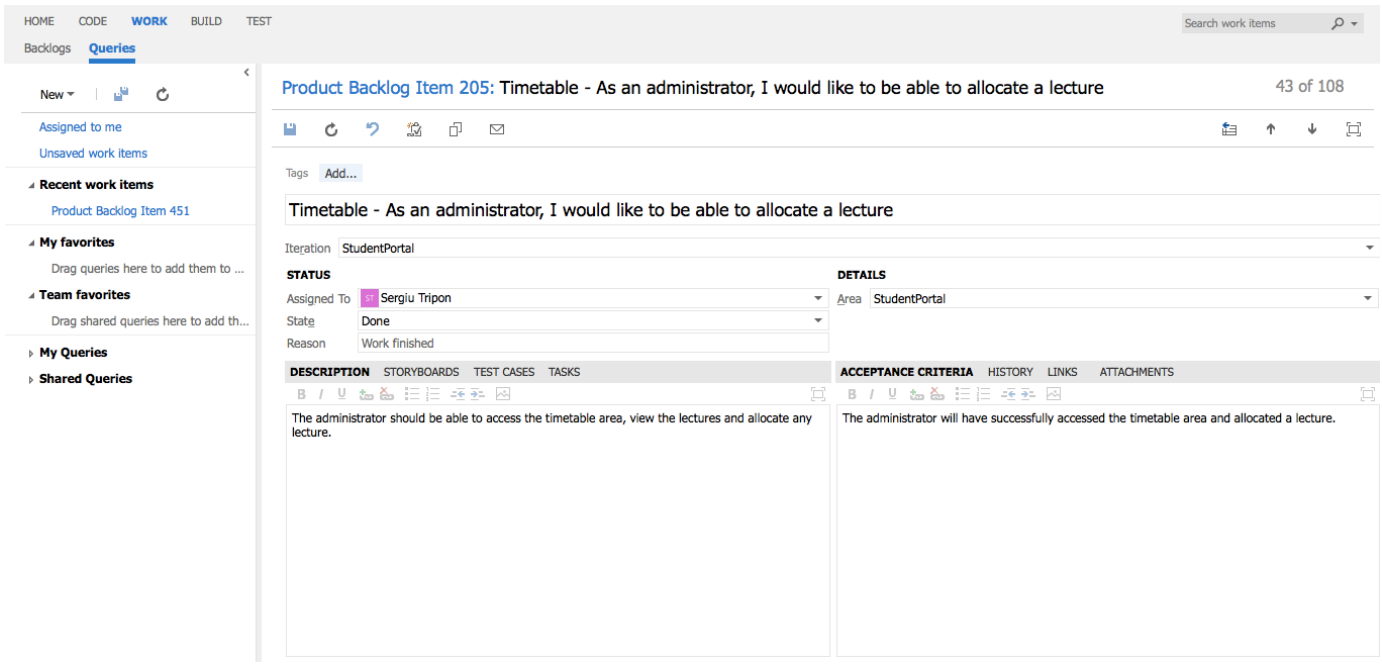
Timetable - detailed view of requirement



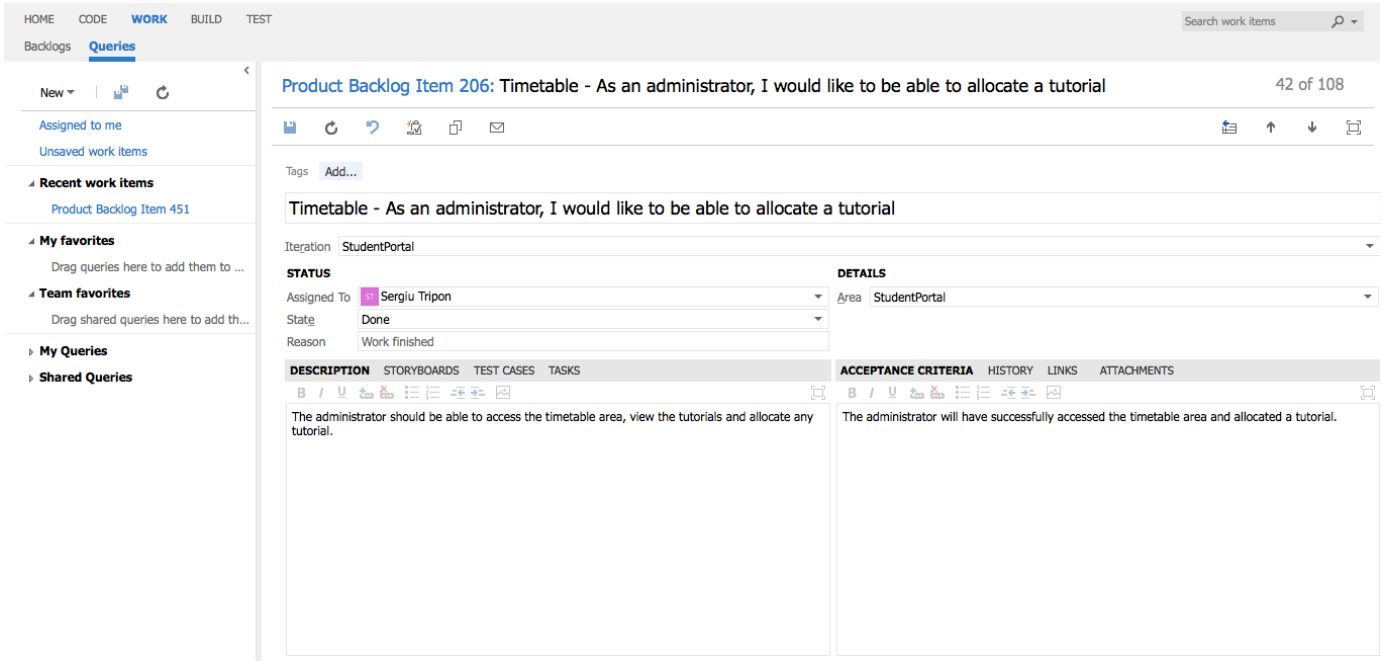
Timetable - detailed view of requirement



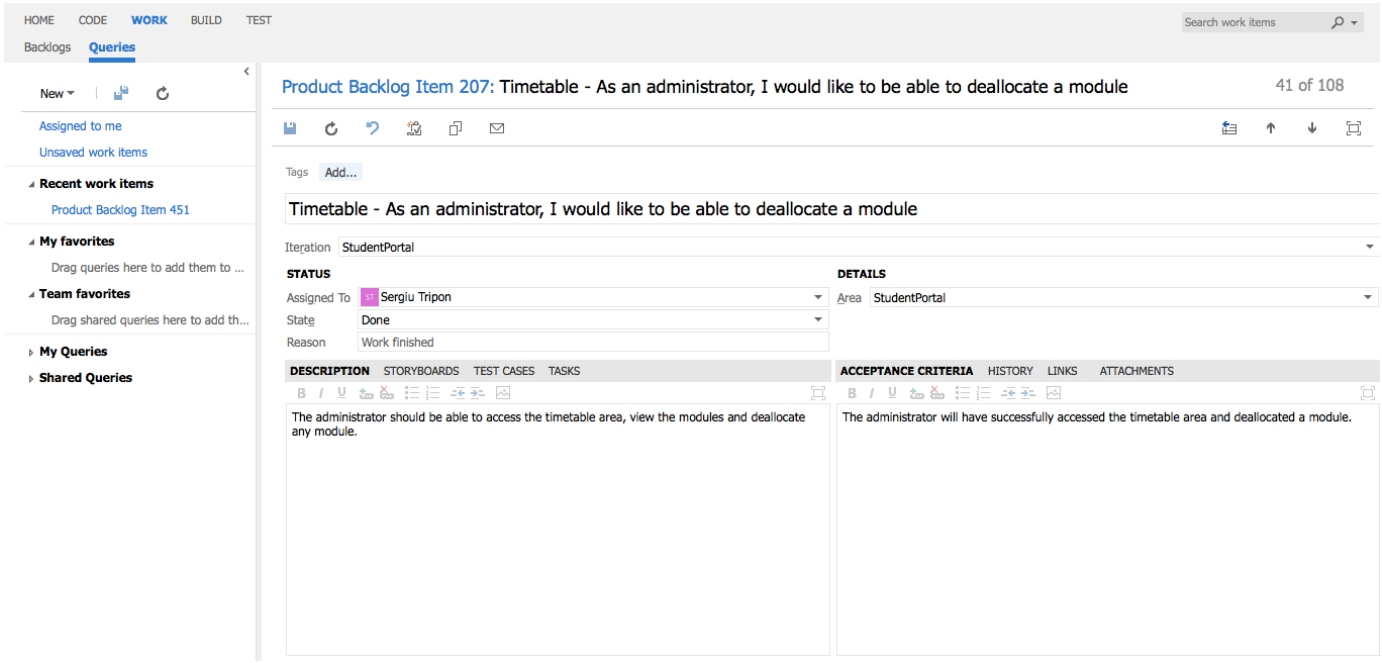
Timetable - detailed view of requirement



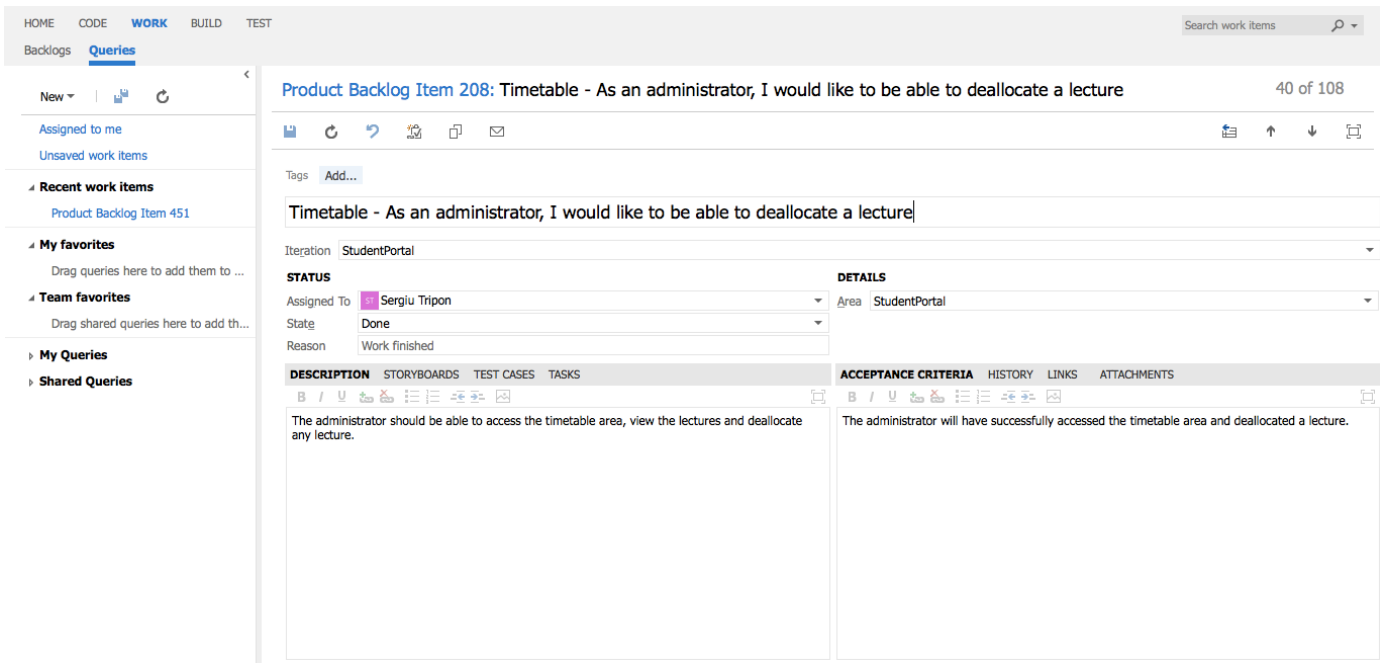
Timetable - detailed view of requirement



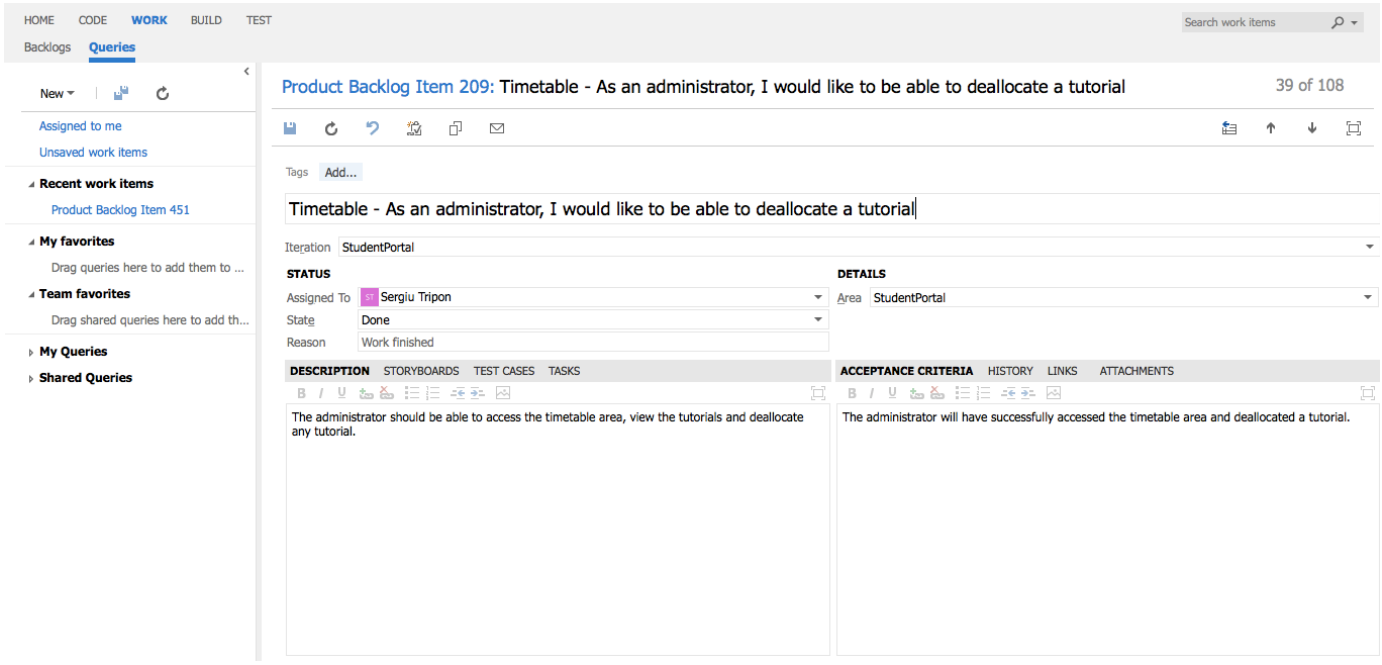
Timetable - detailed view of requirement



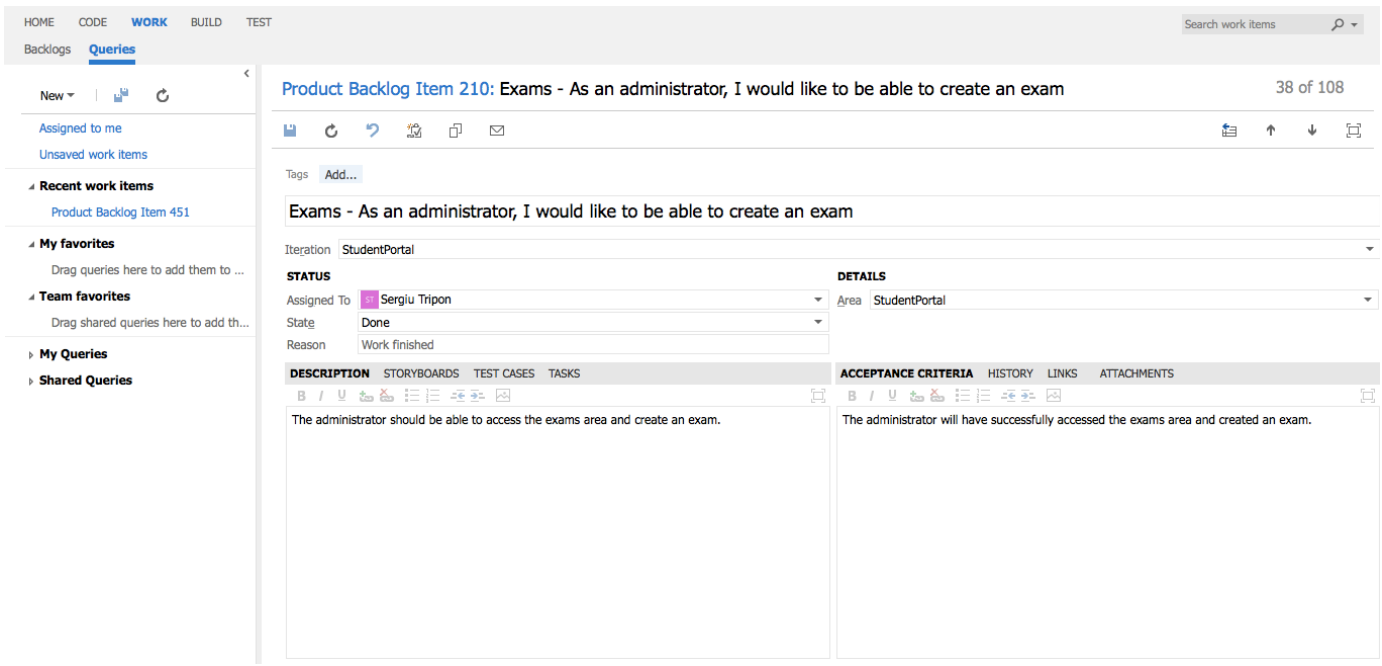
Timetable - detailed view of requirement



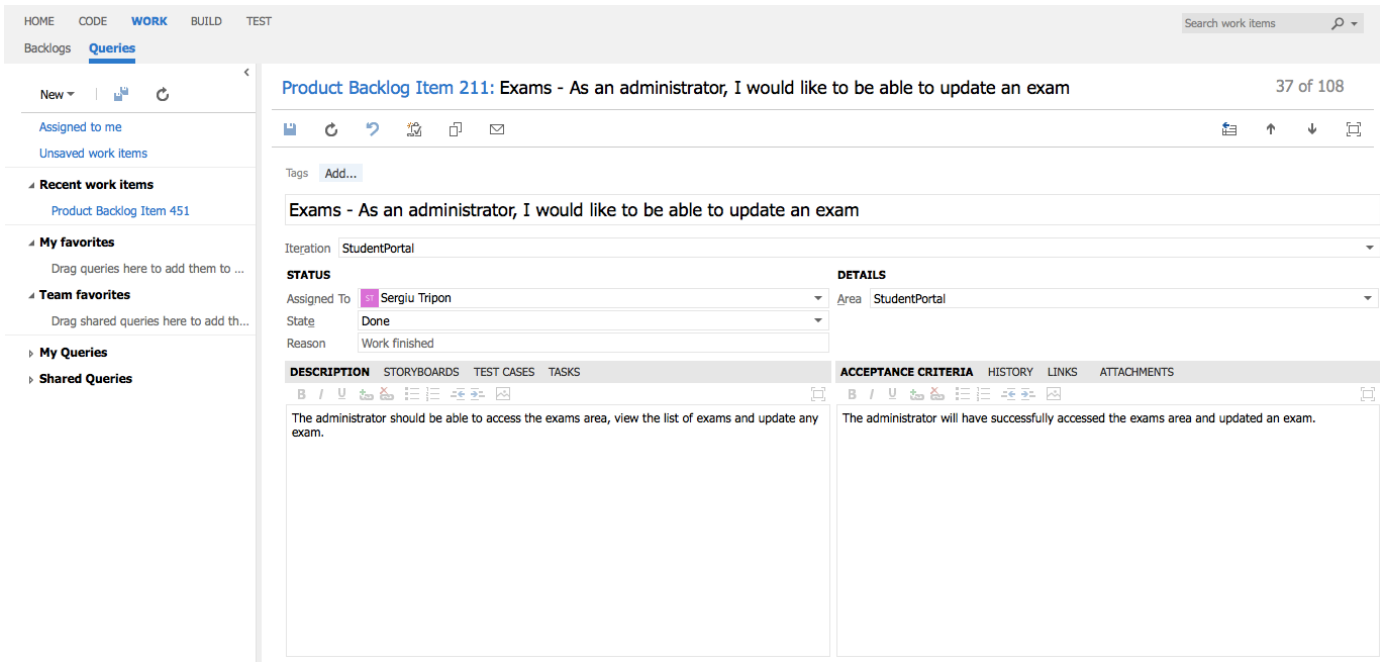
Timetable - detailed view of requirement



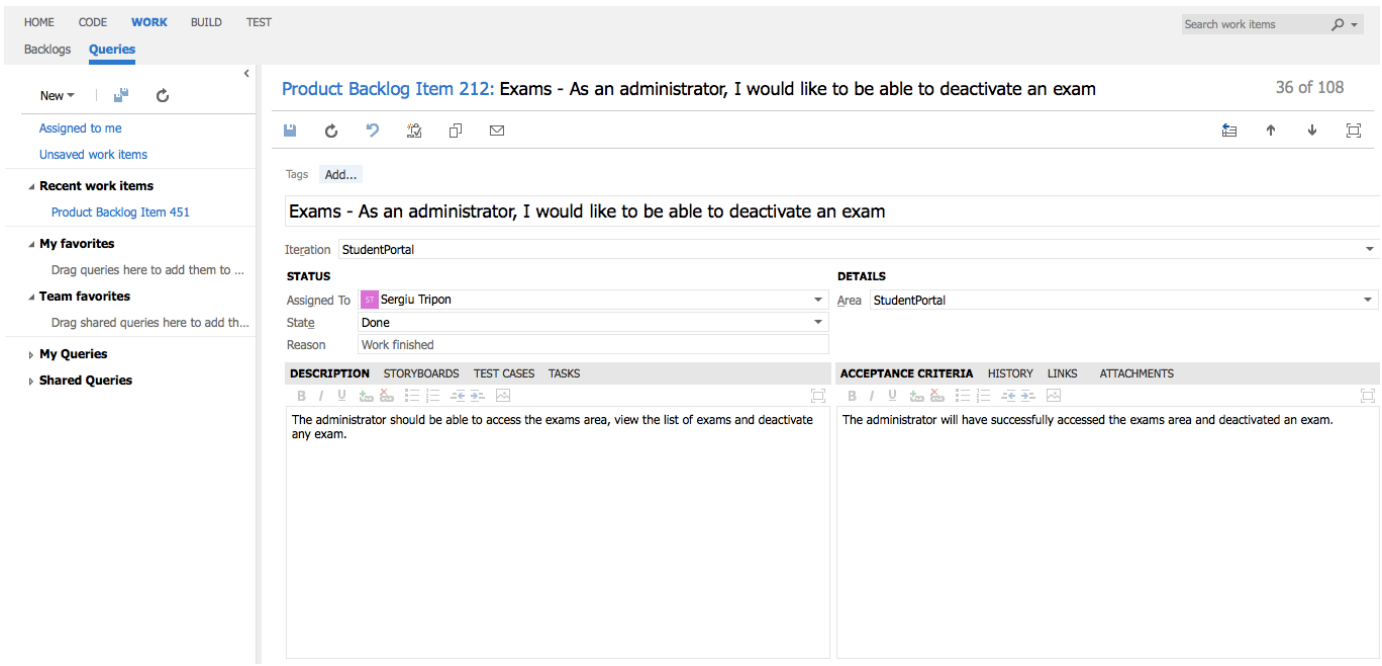
Exams - detailed view of requirement



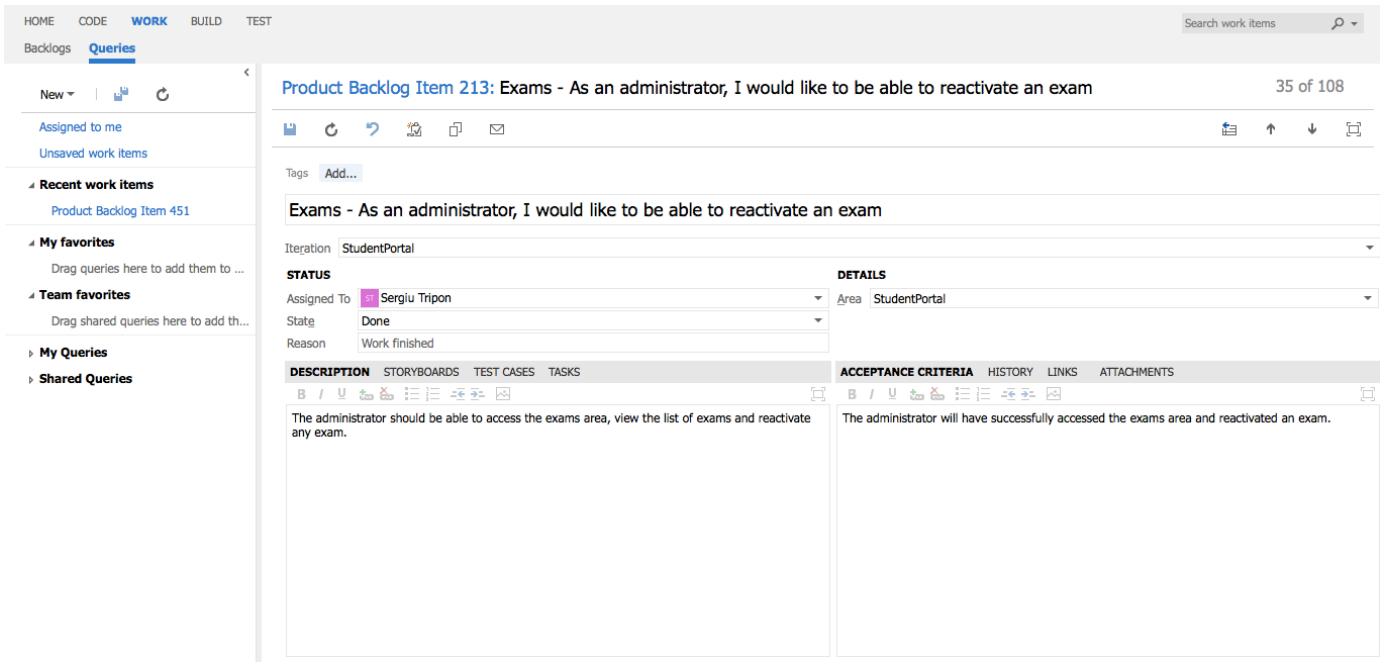
Exams - detailed view of requirement



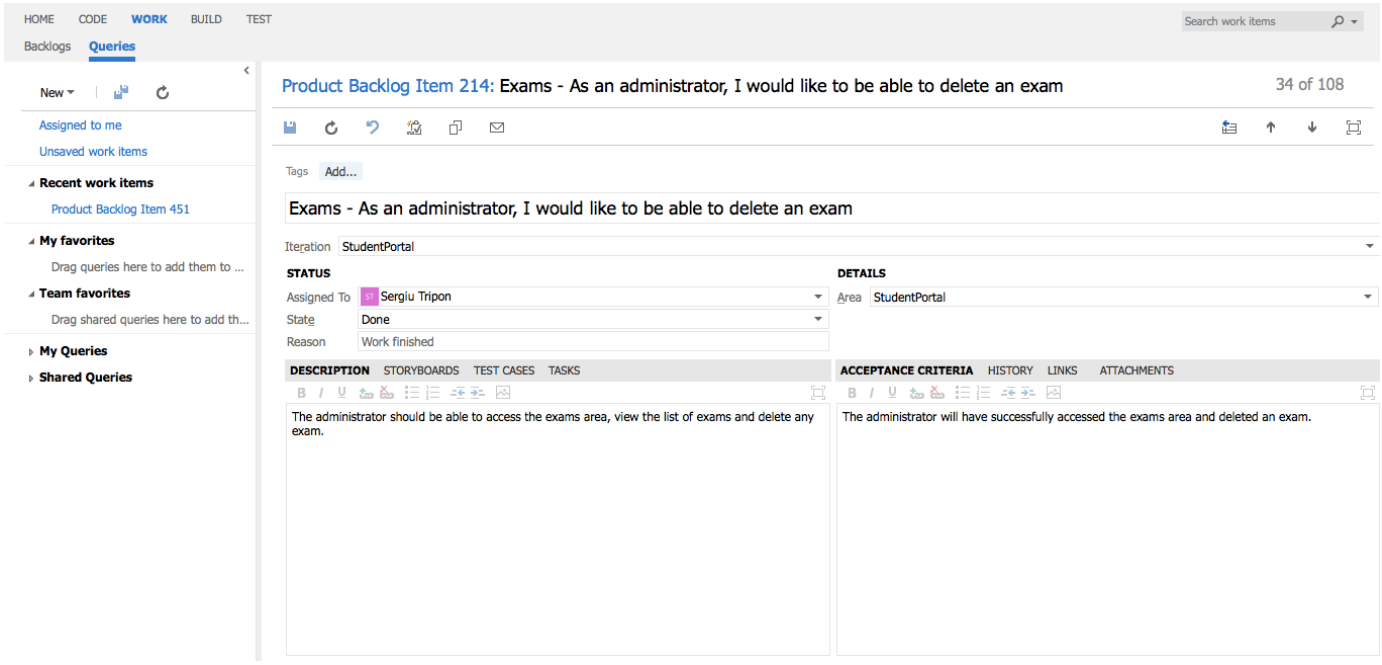
Exams - detailed view of requirement



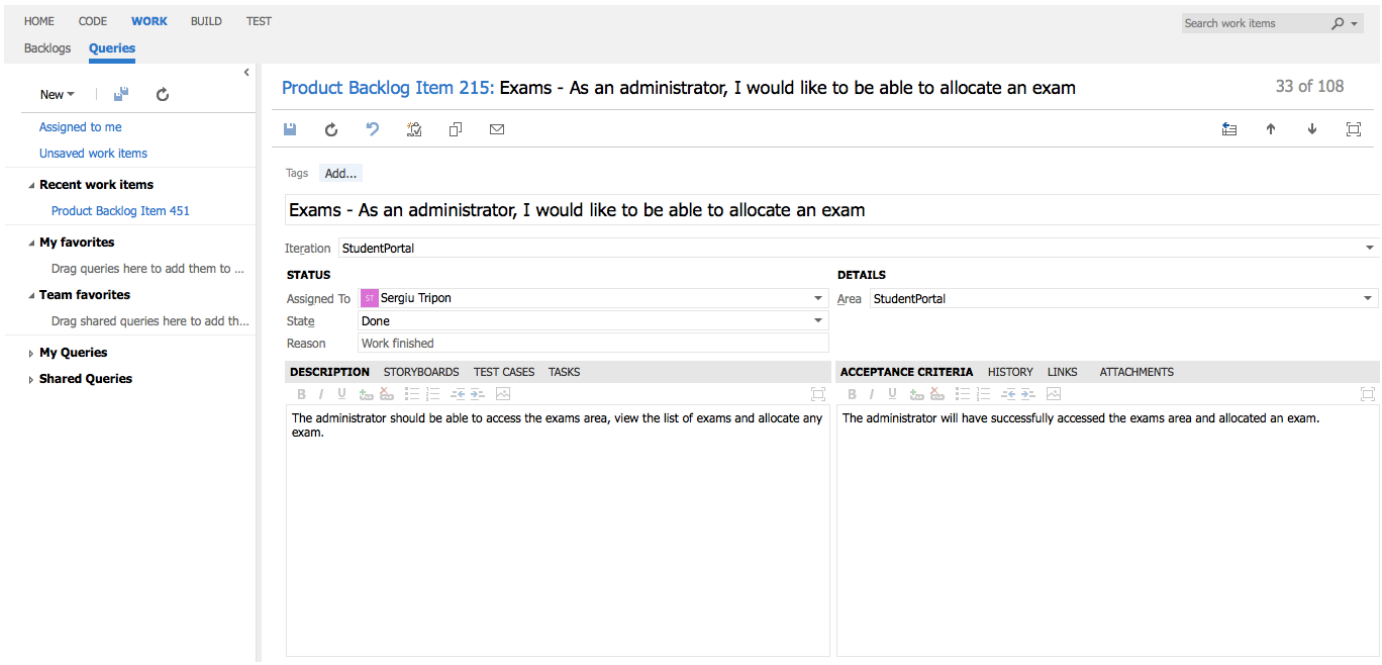
Exams - detailed view of requirement



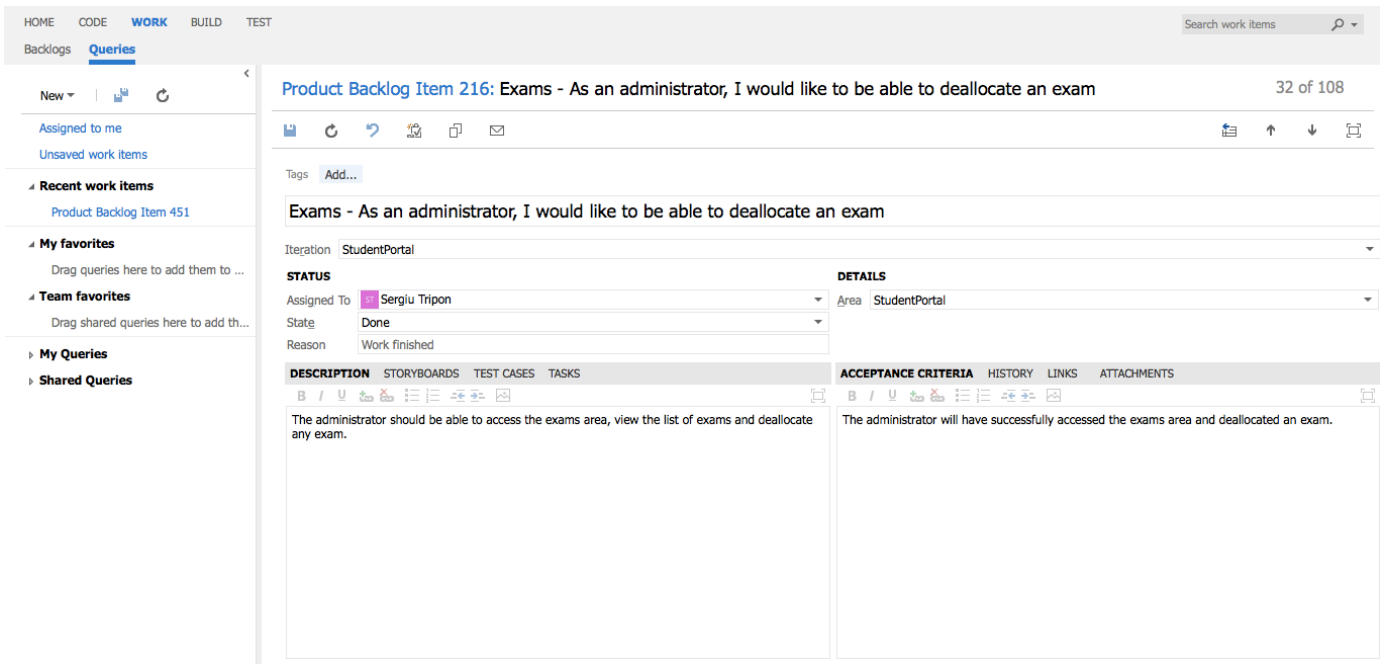
Exams - detailed view of requirement



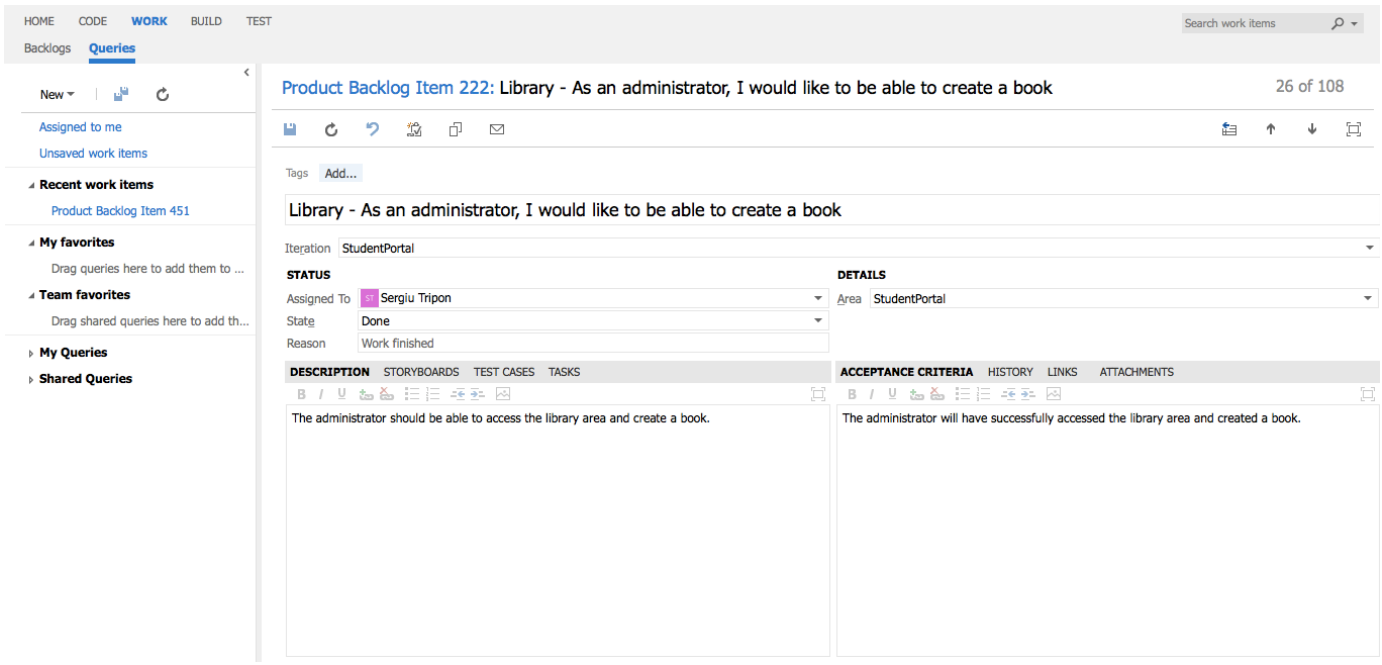
Exams - detailed view of requirement



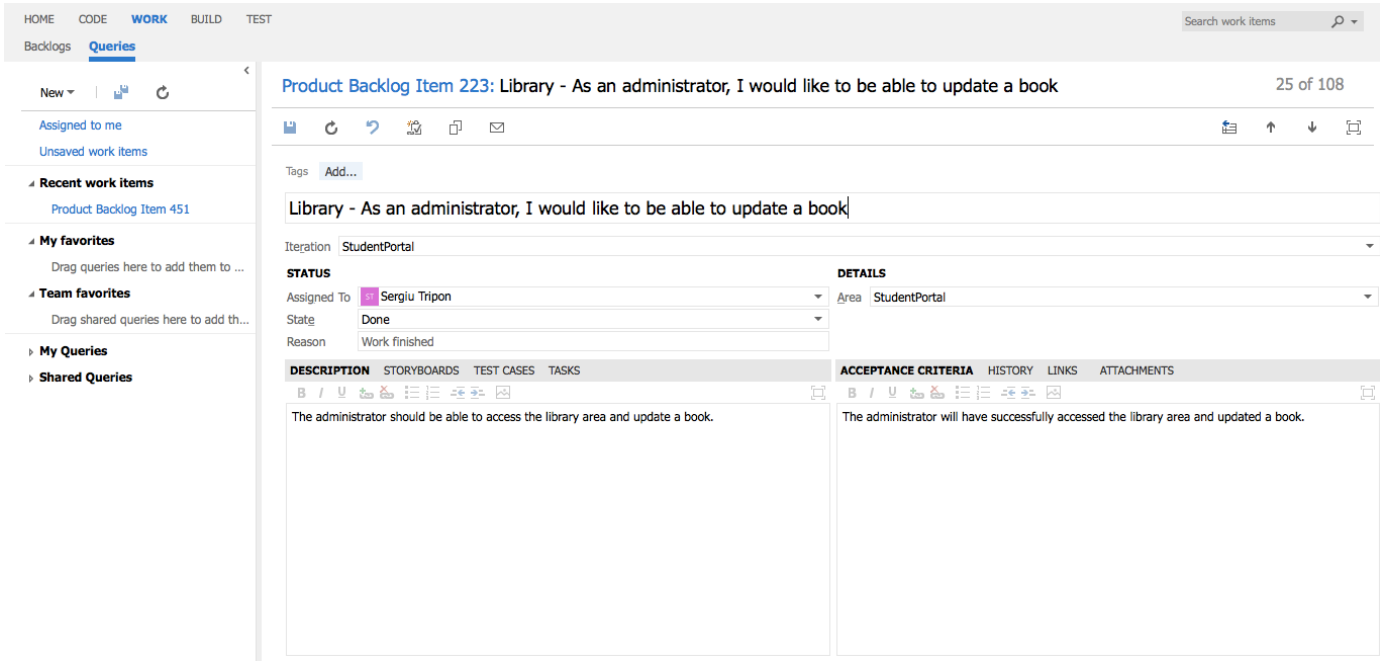
Exams - detailed view of requirement



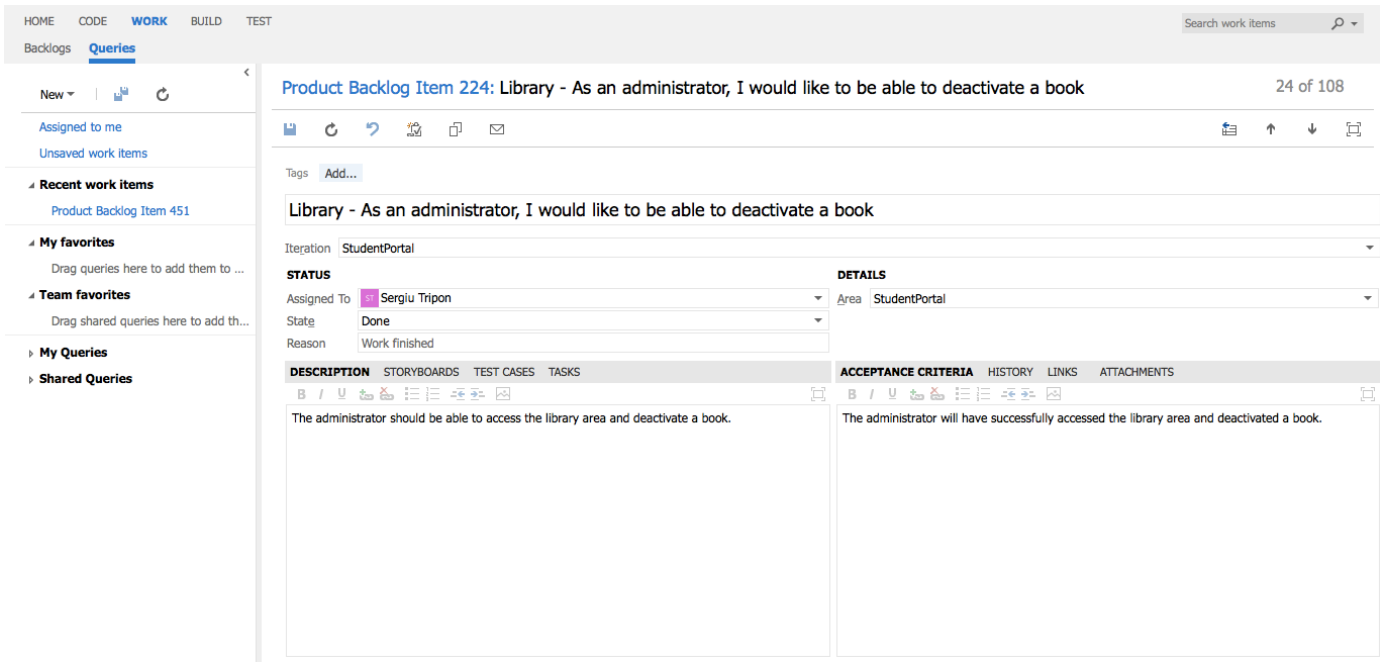
Library - detailed view of requirement



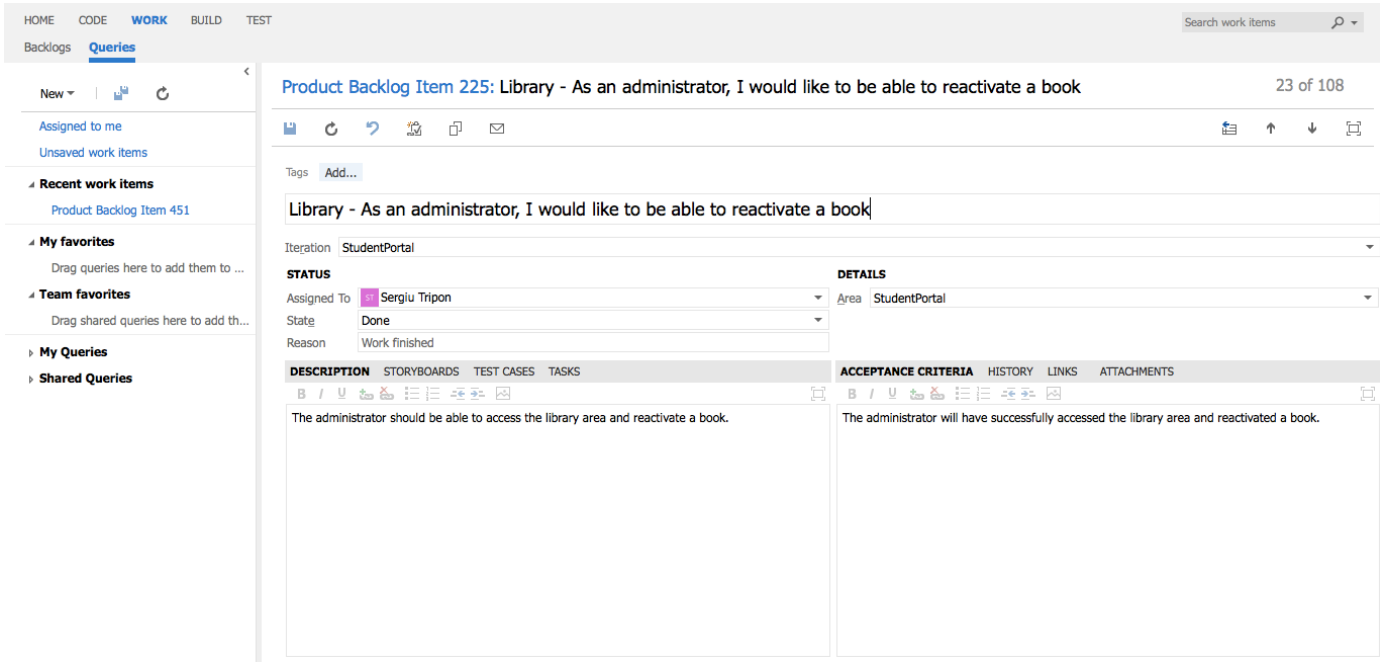
Library - detailed view of requirement



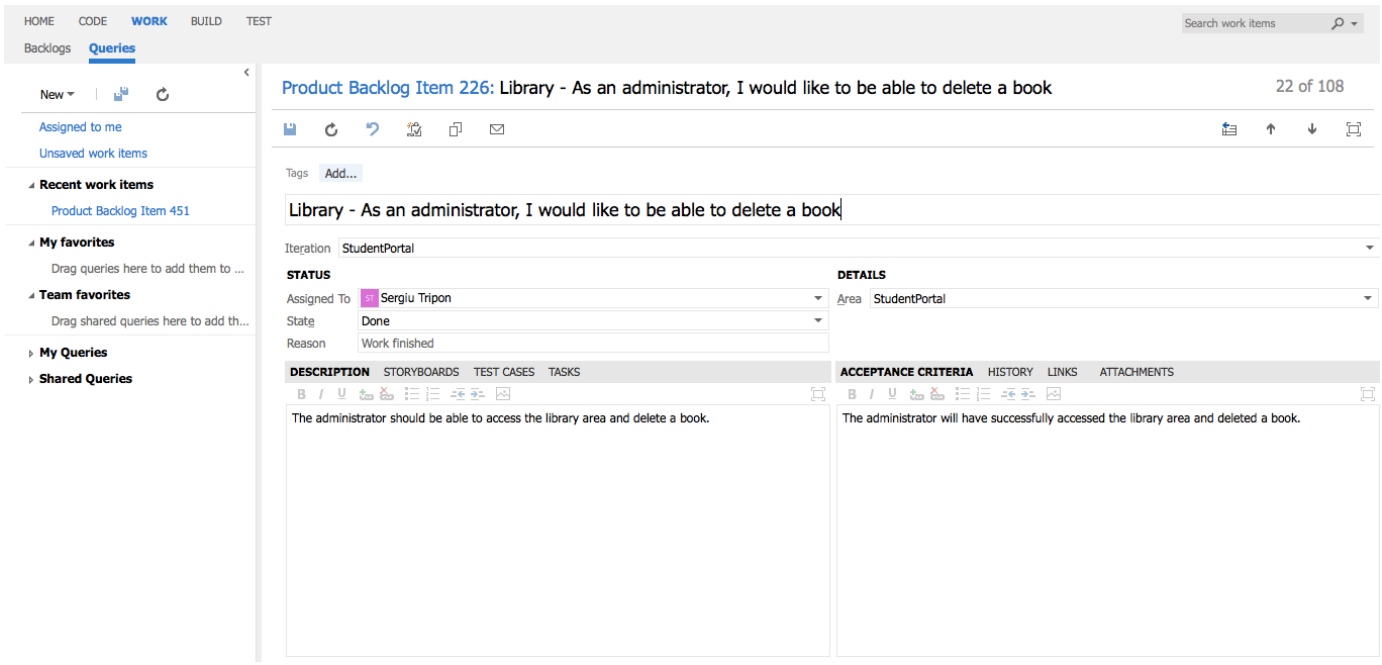
Library - detailed view of requirement



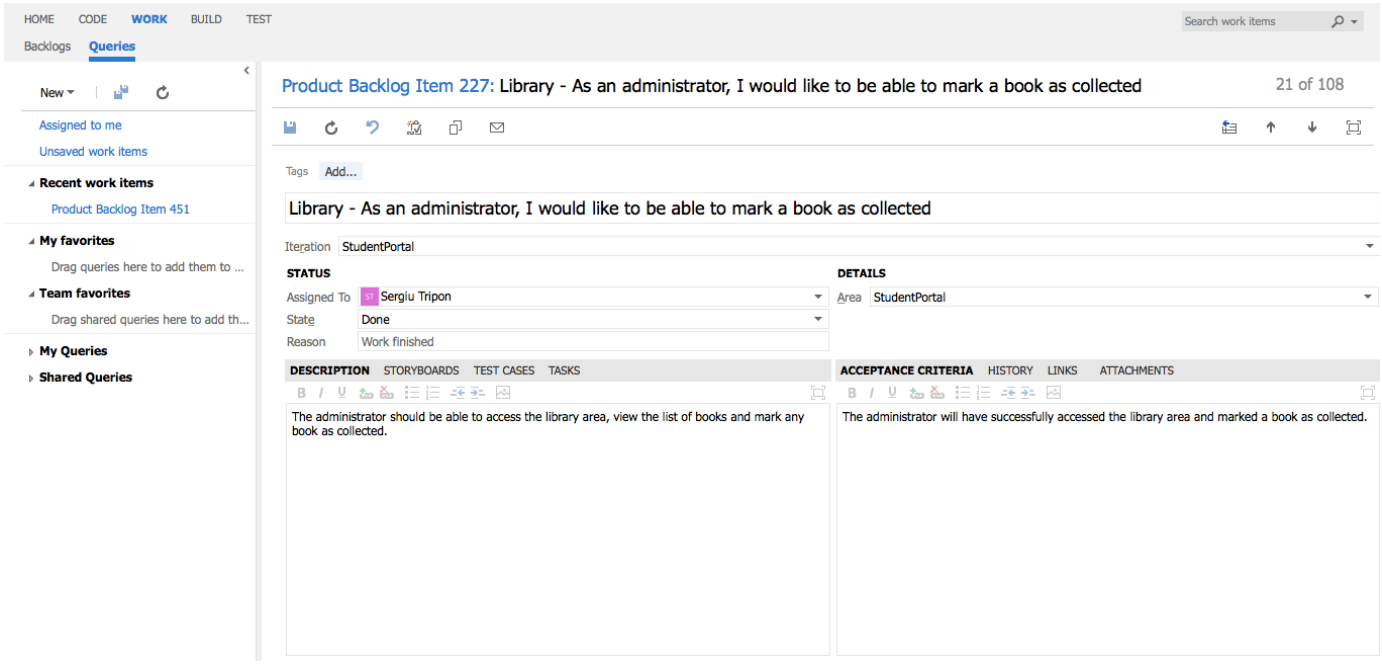
Library - detailed view of requirement



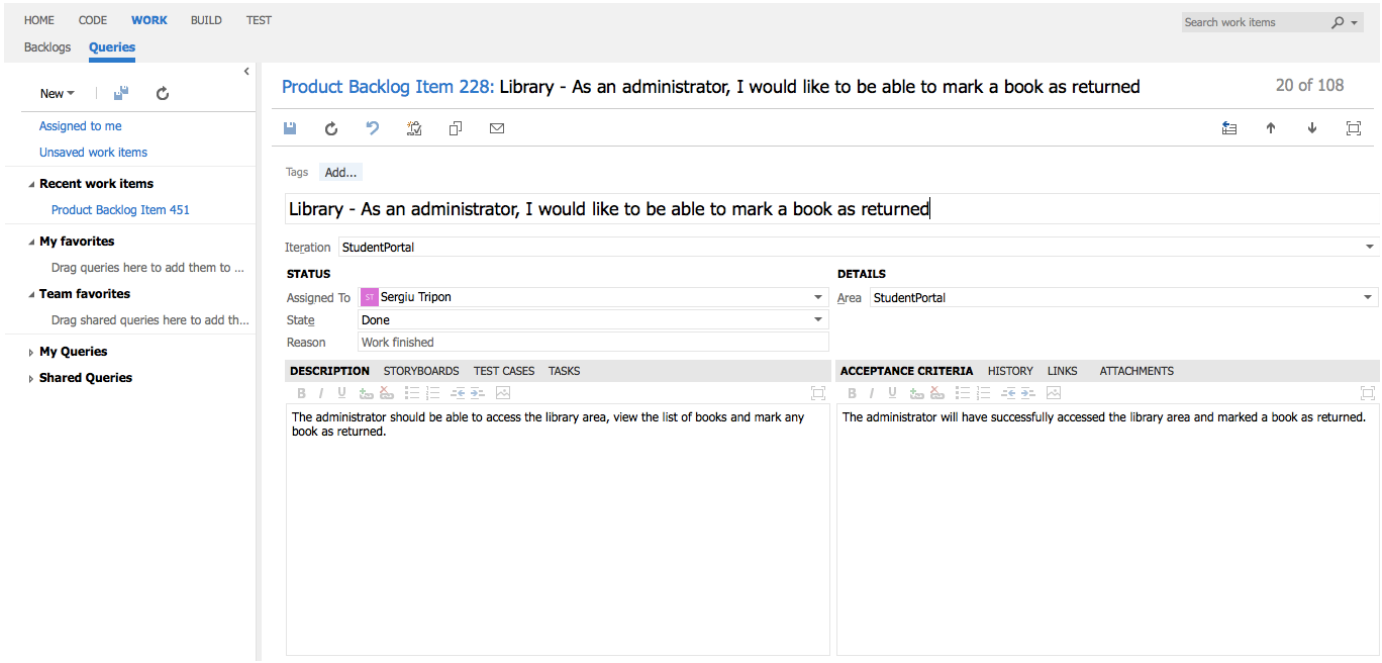
Library - detailed view of requirement



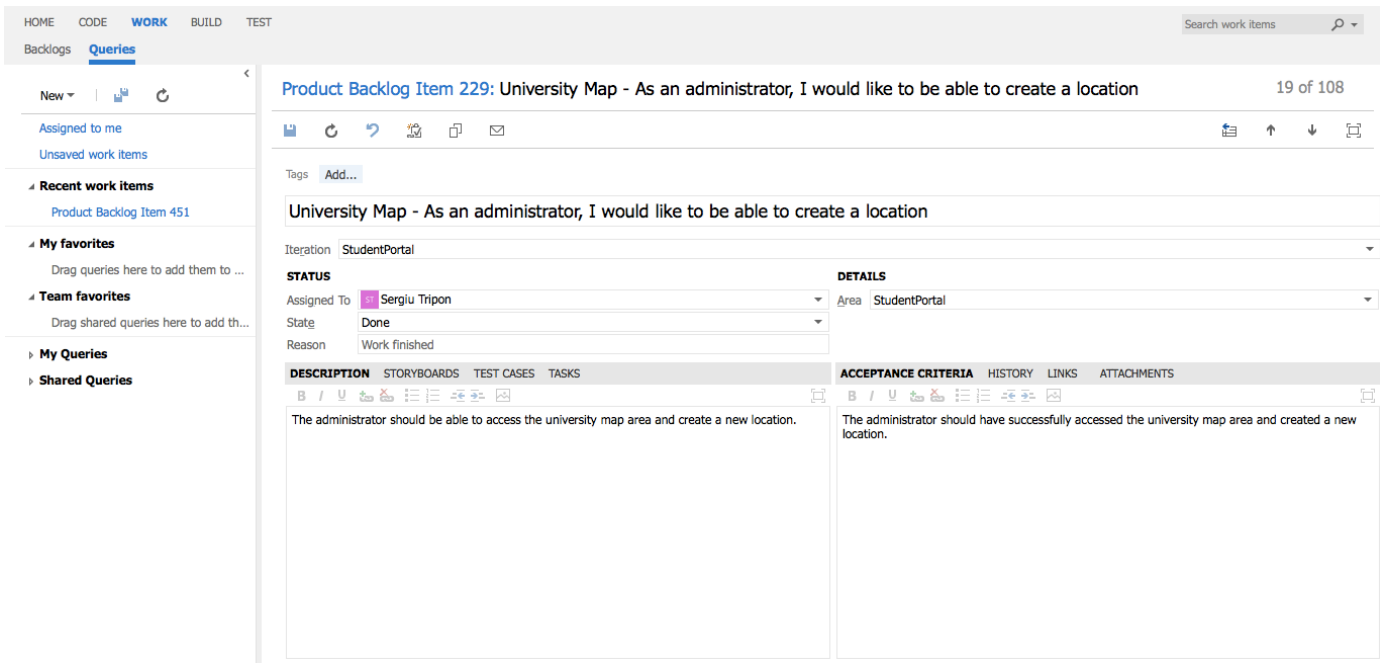
Library - detailed view of requirement



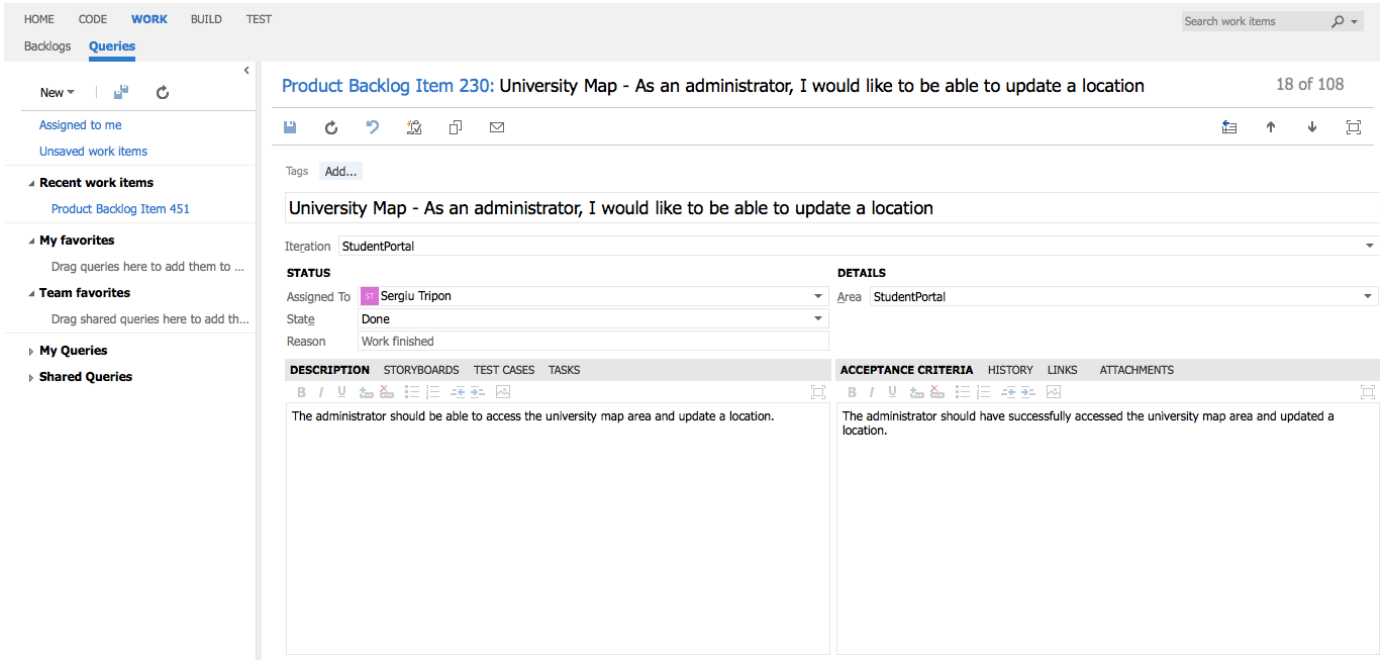
Library - detailed view of requirement



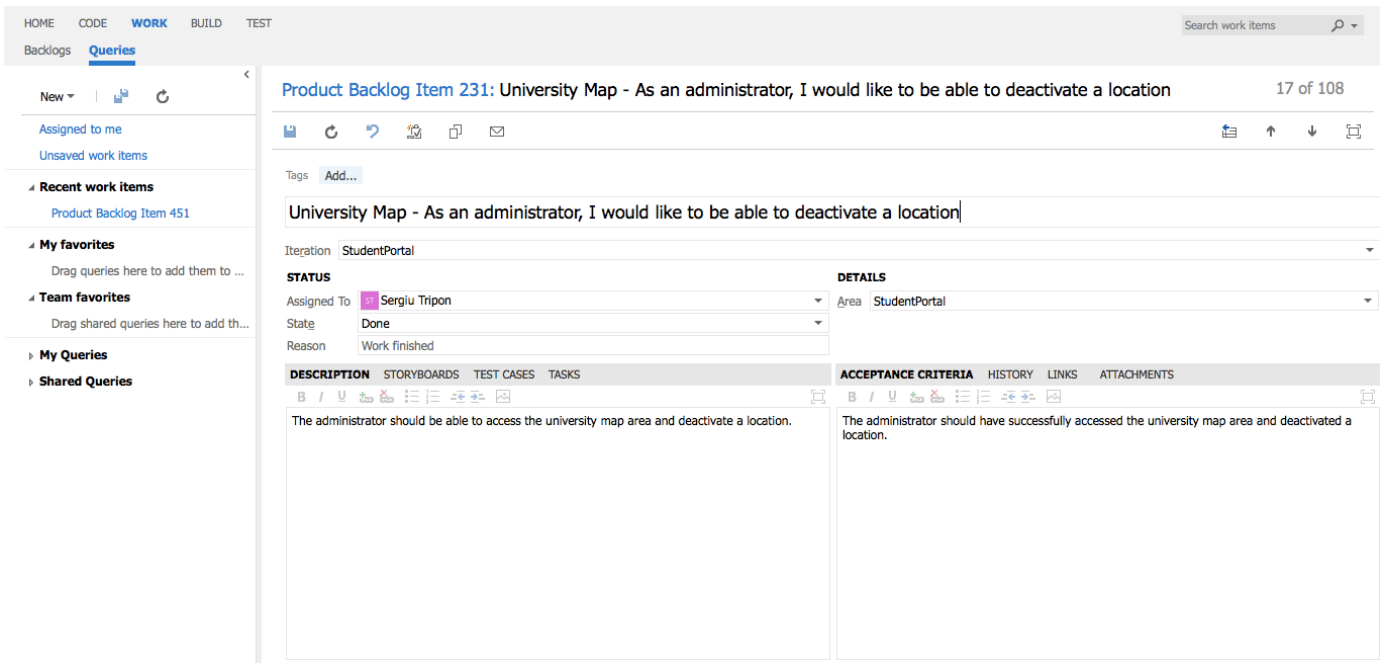
University Map - detailed view of requirement



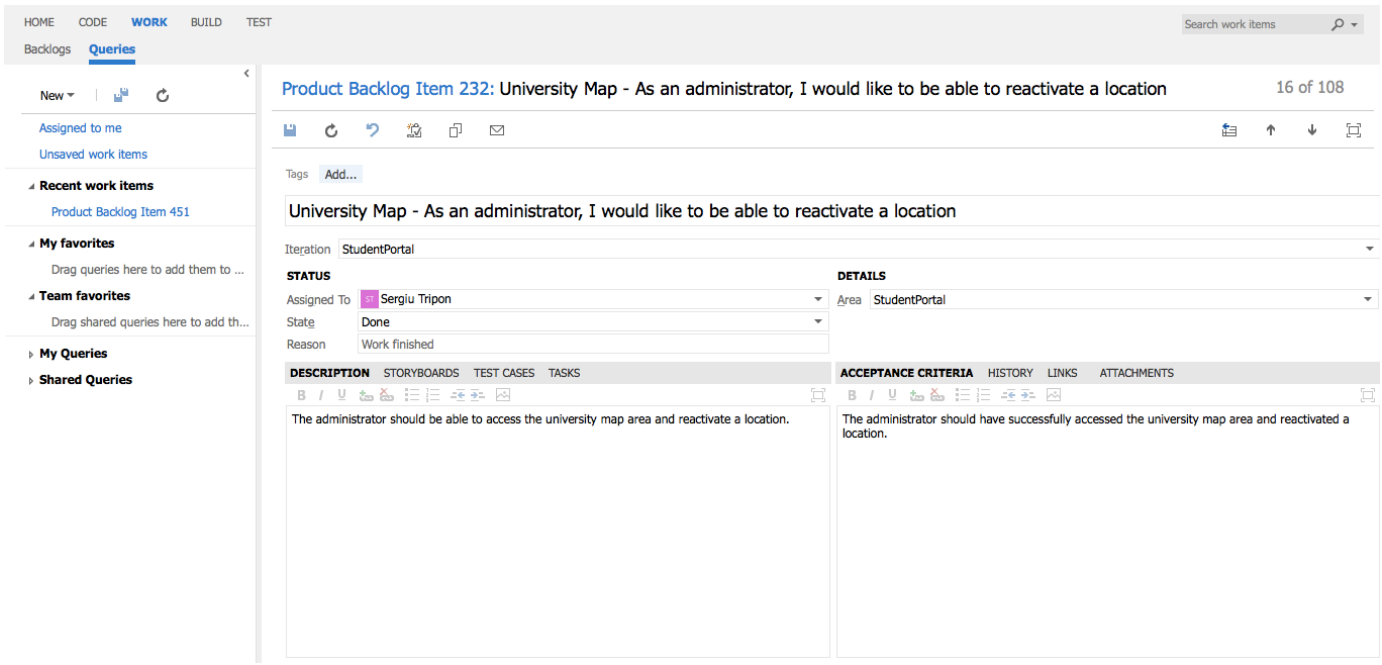
University Map - detailed view of requirement



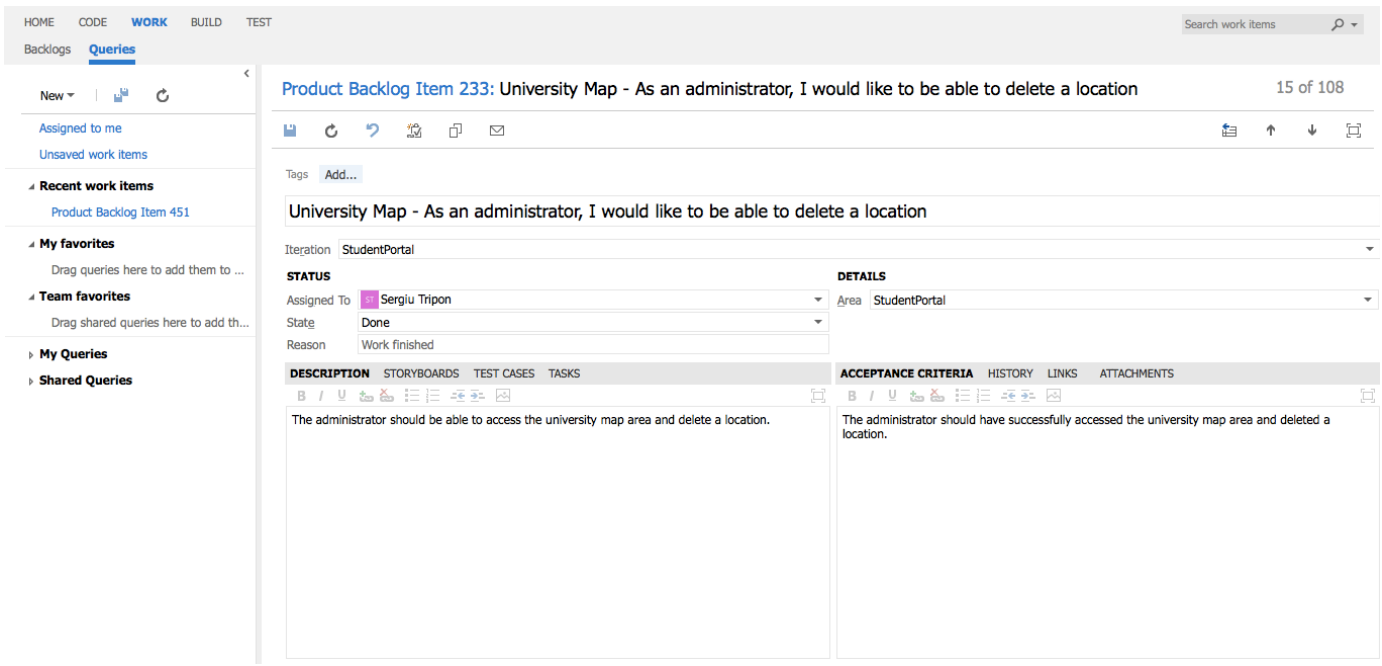
University Map - detailed view of requirement



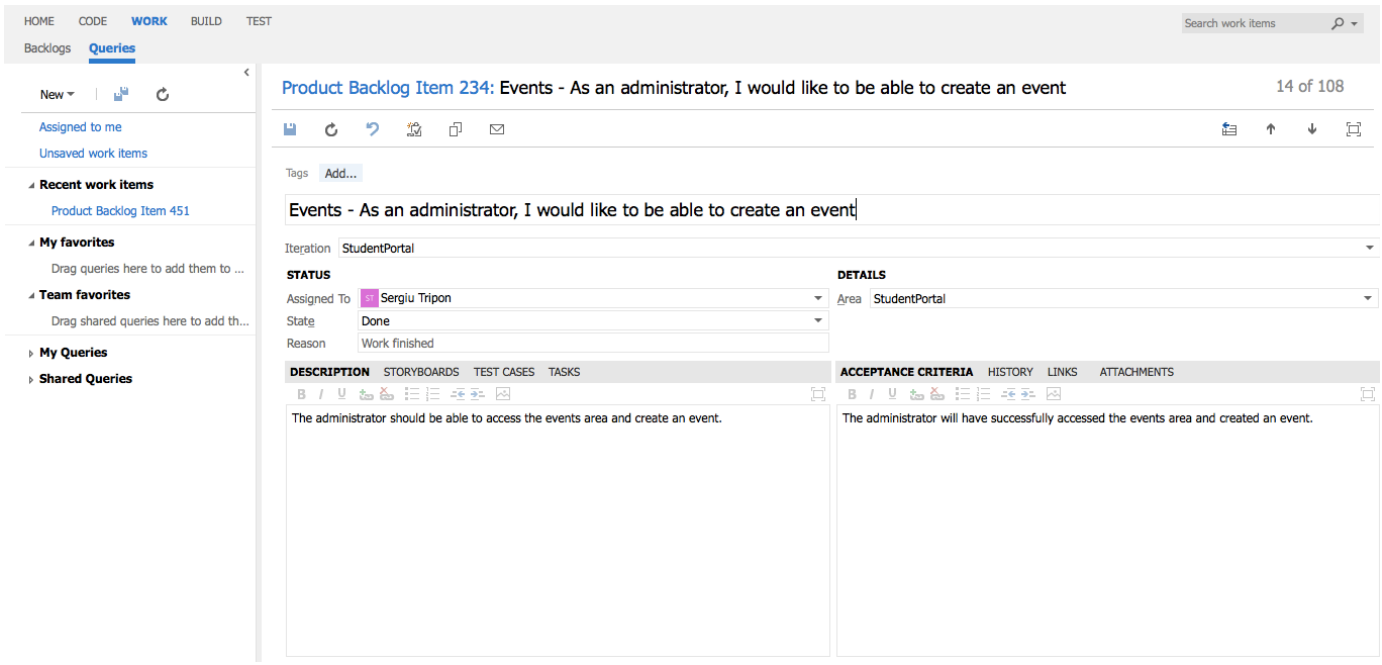
University Map - detailed view of requirement



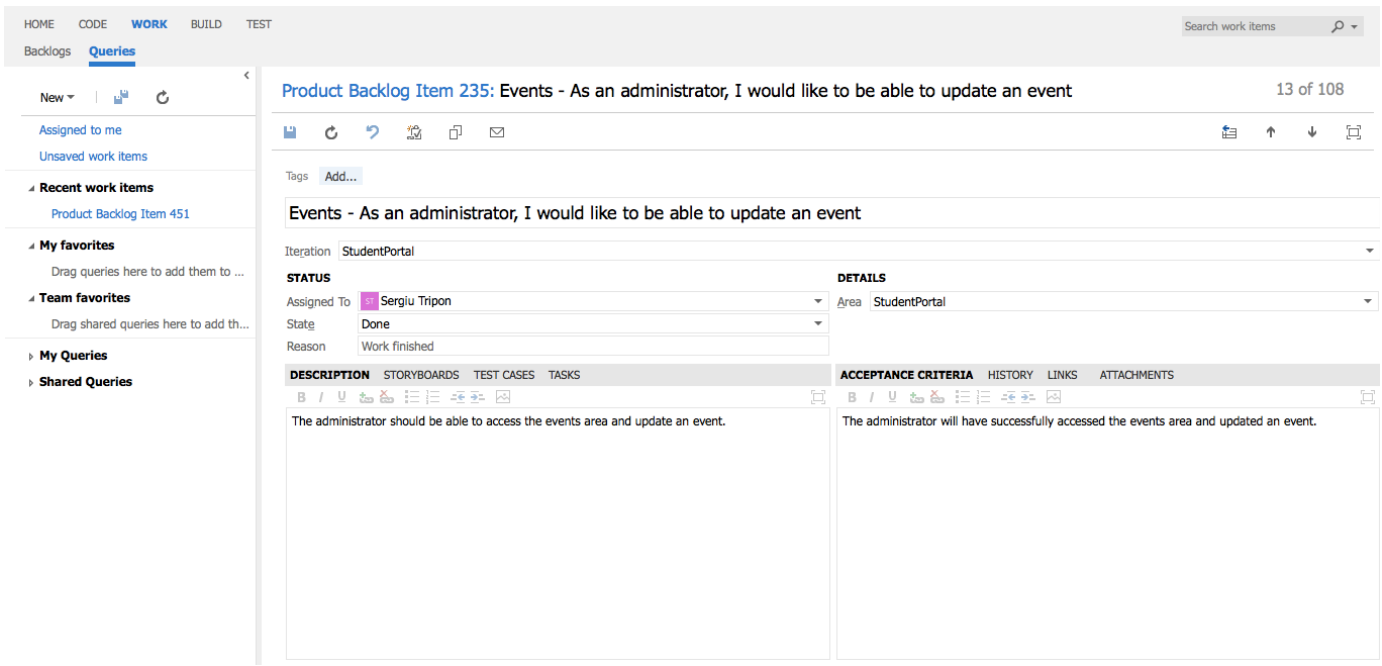
University Map - detailed view of requirement



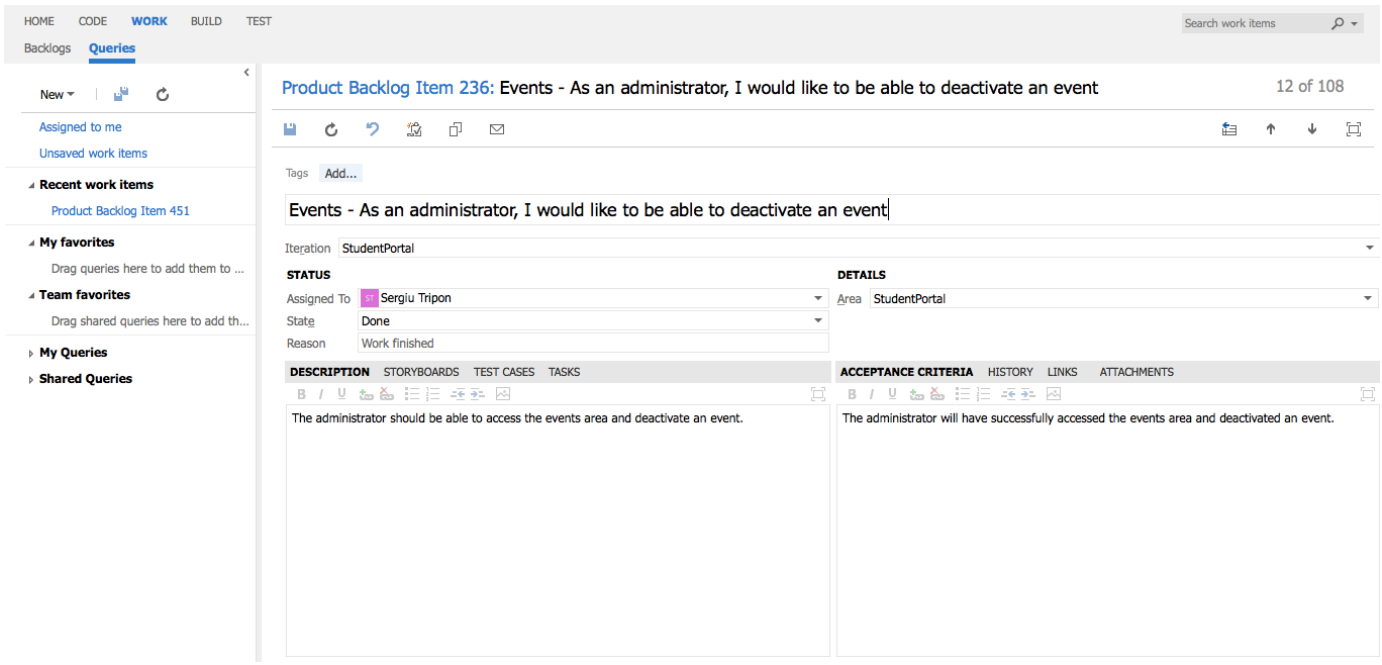
Events - detailed view of requirement



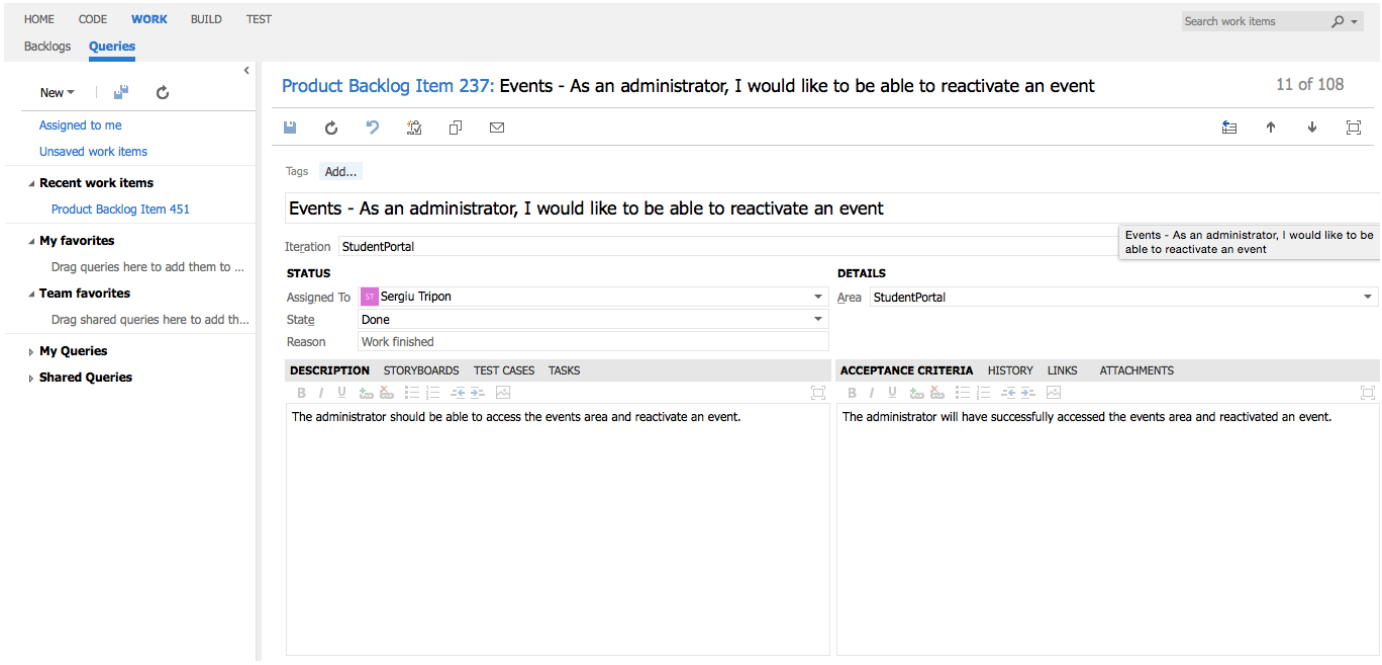
Events - detailed view of requirement



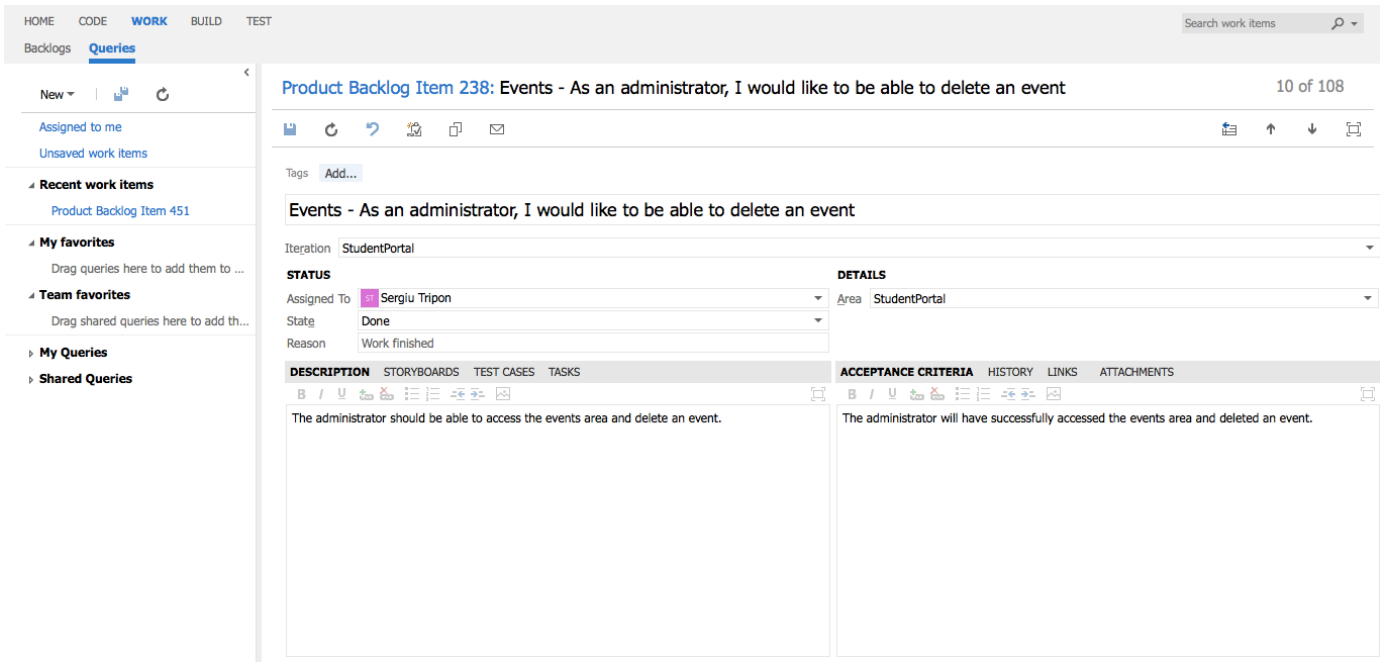
Events - detailed view of requirement



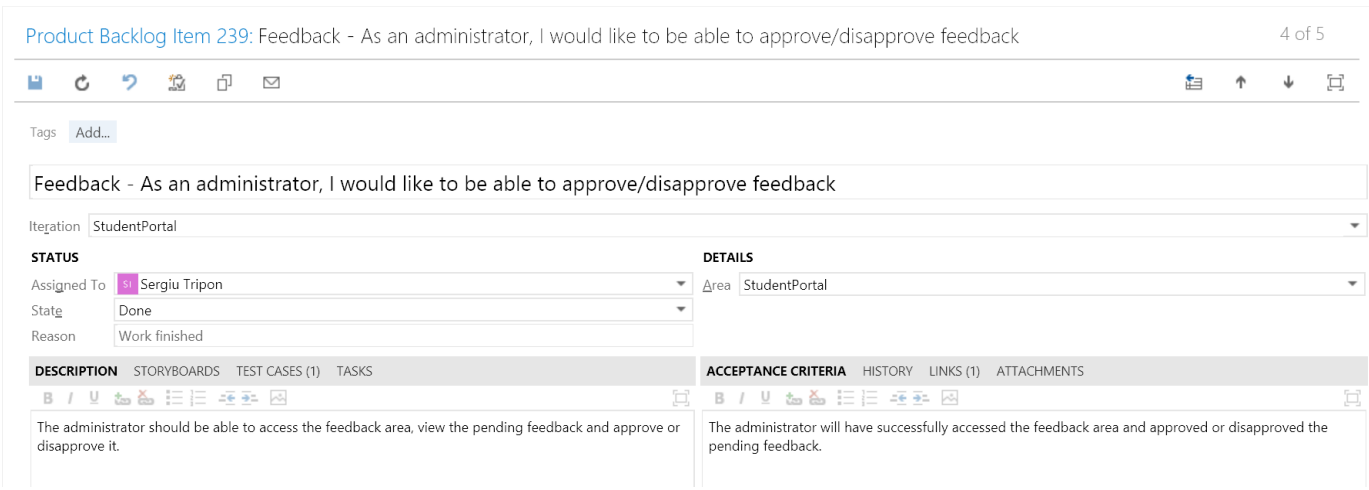
Events - detailed view of requirement



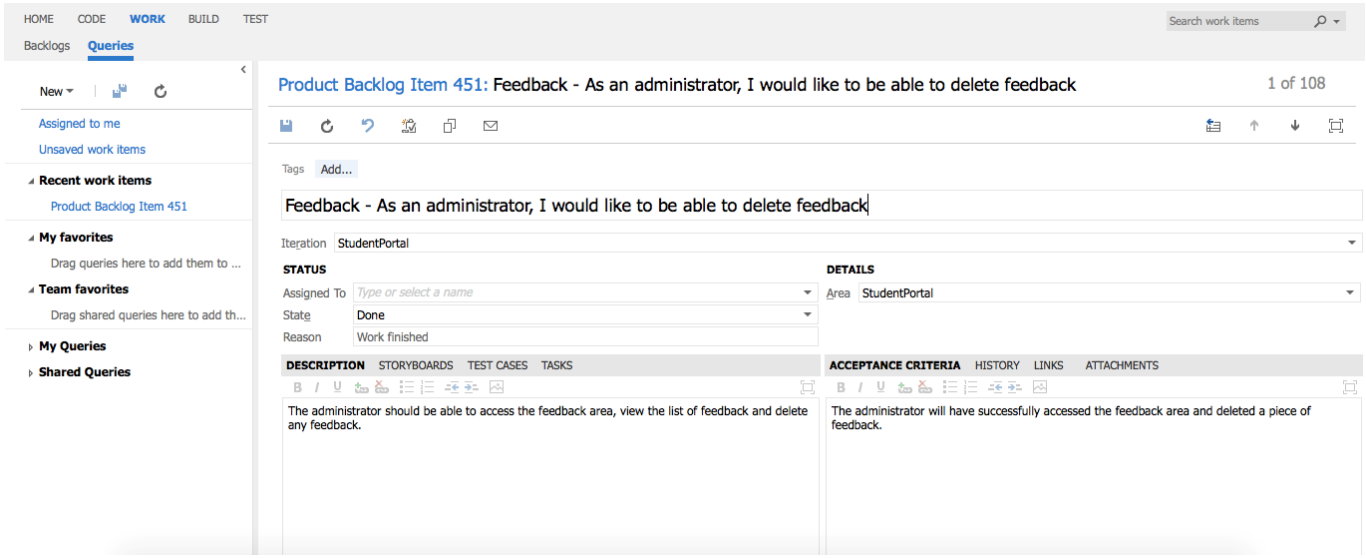
Events - detailed view of requirement



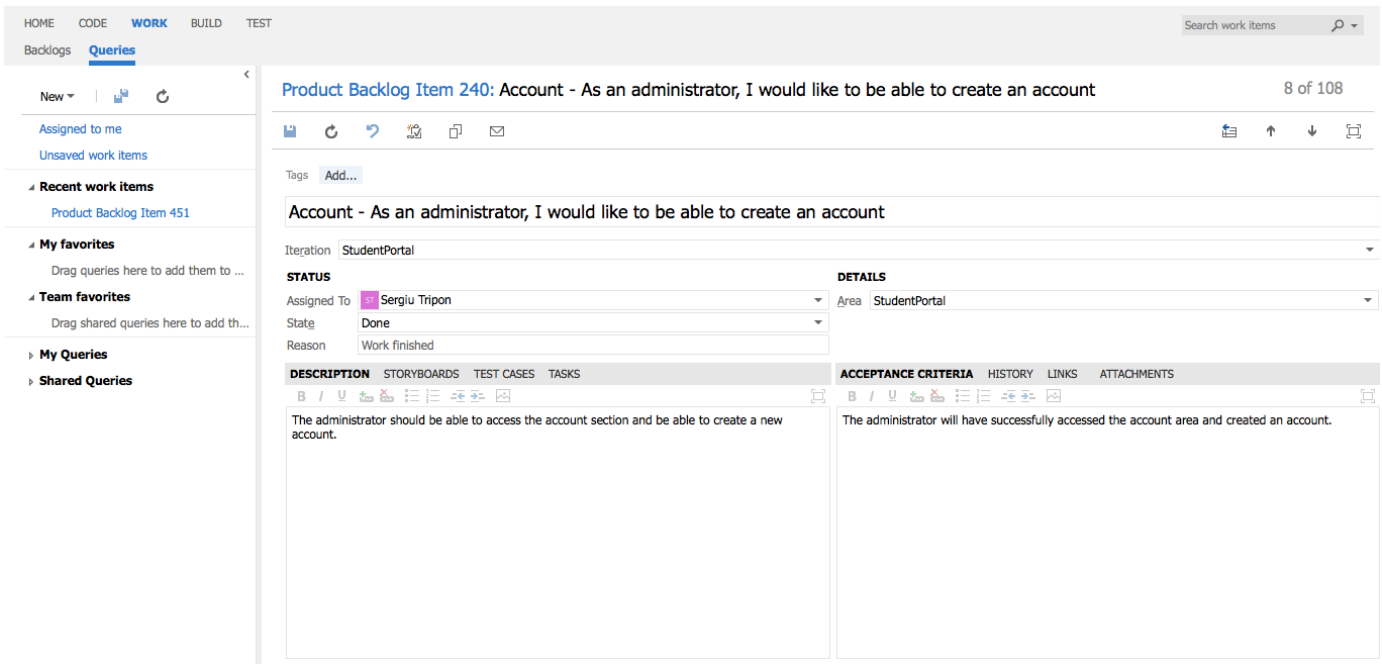
Feedback - detailed view of requirement



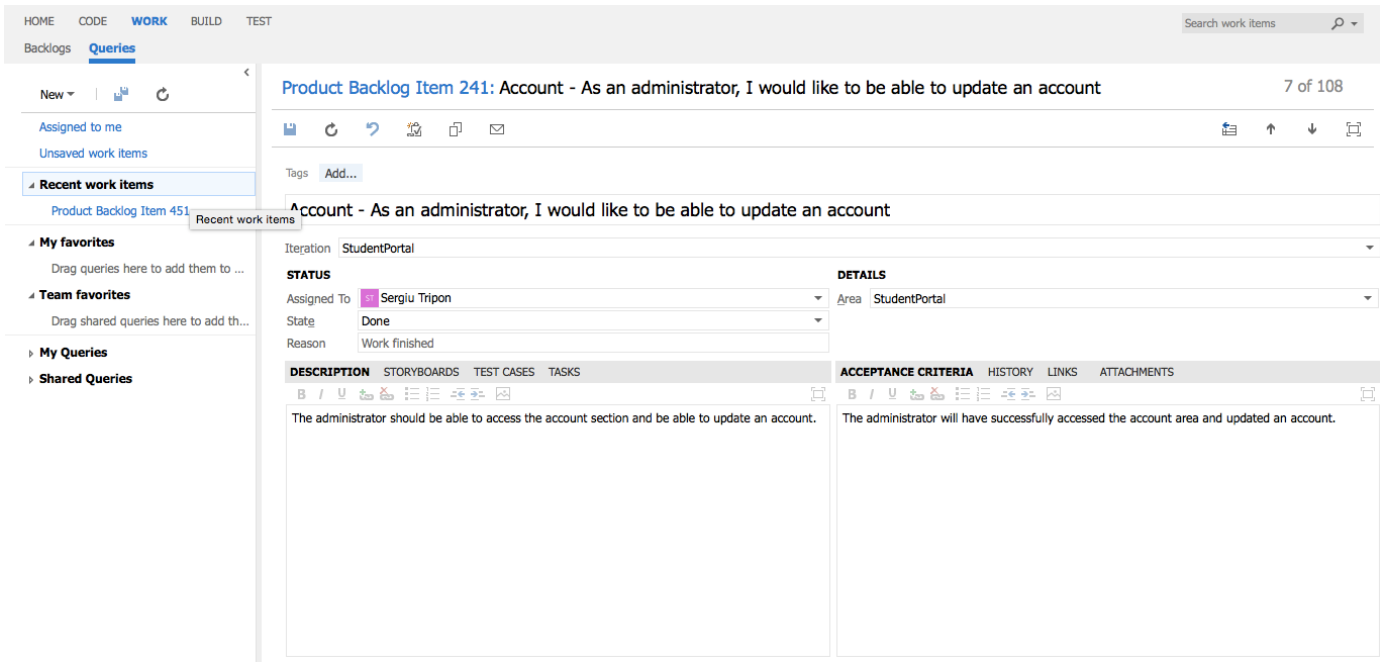
Feedback - detailed view of requirement



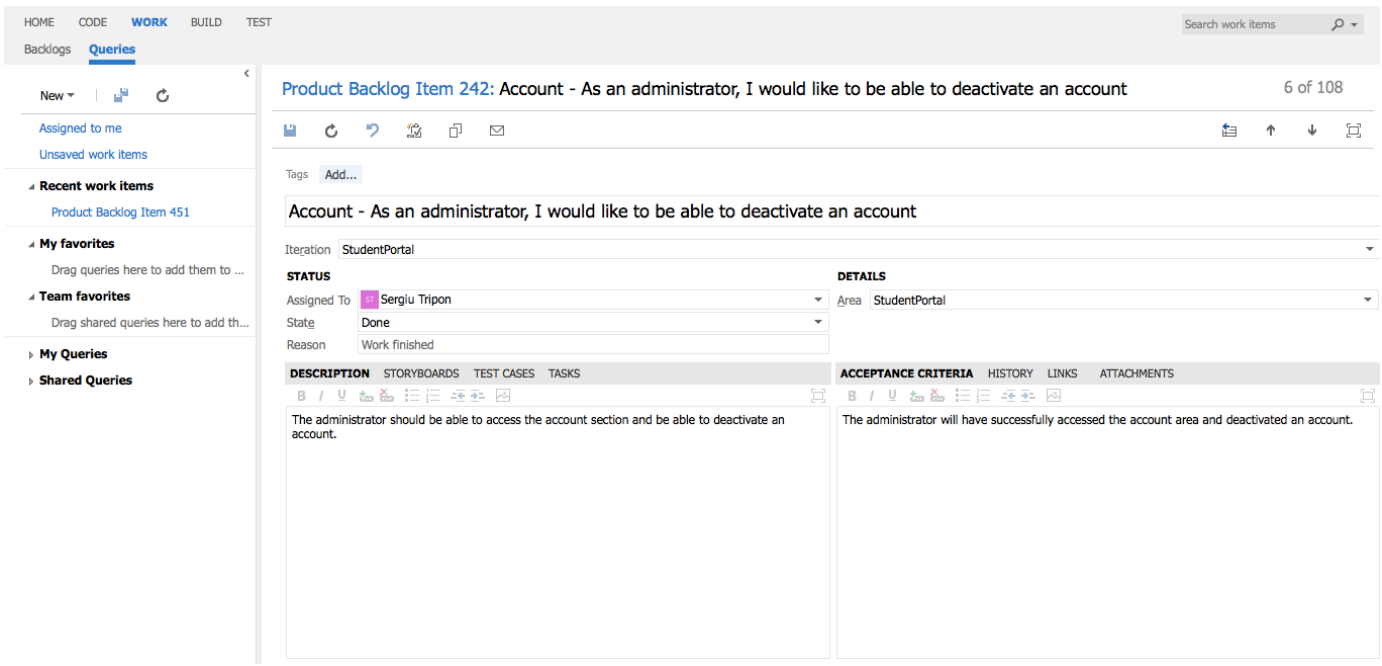
Account - detailed view of requirement



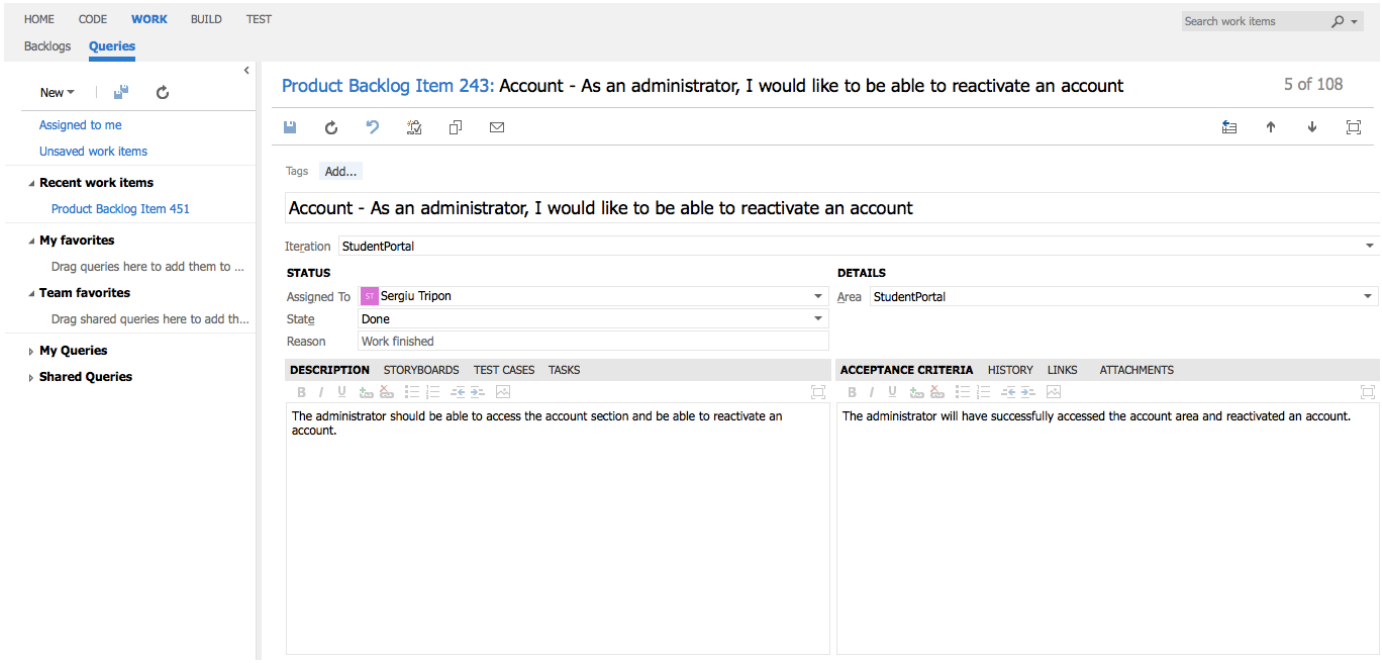
Account - detailed view of requirement



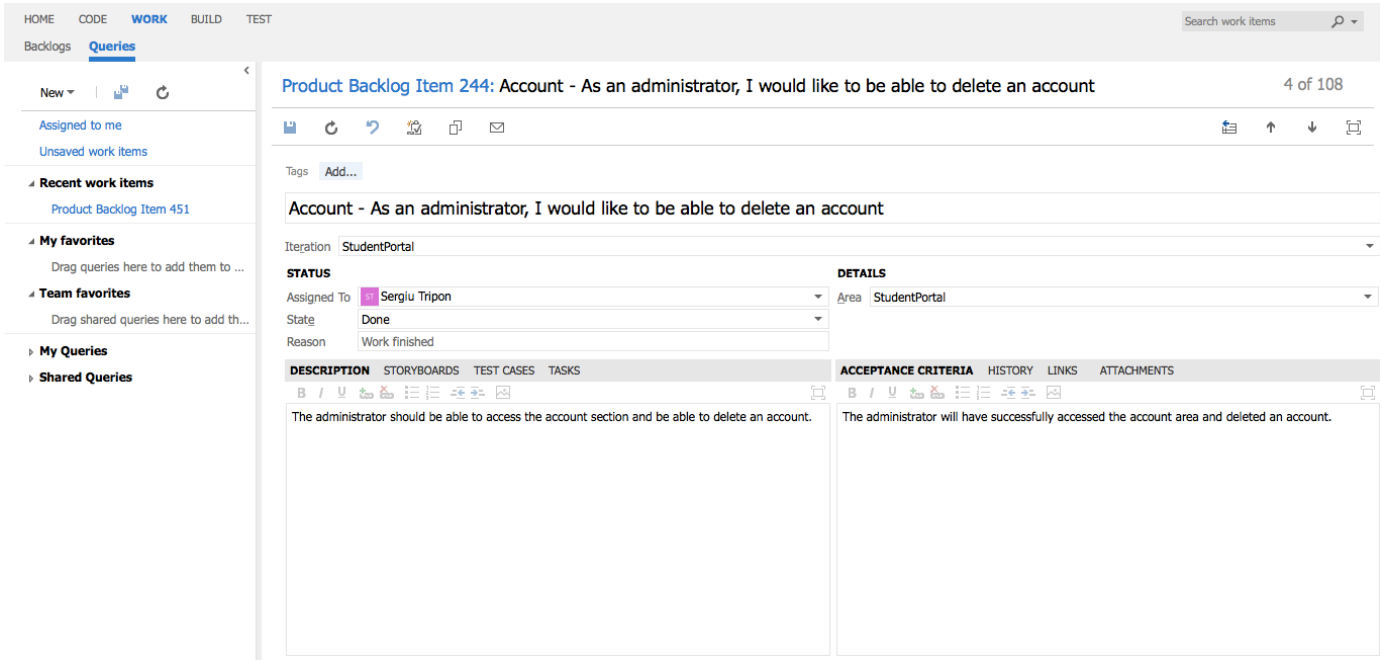
Account - detailed view of requirement



Account - detailed view of requirement



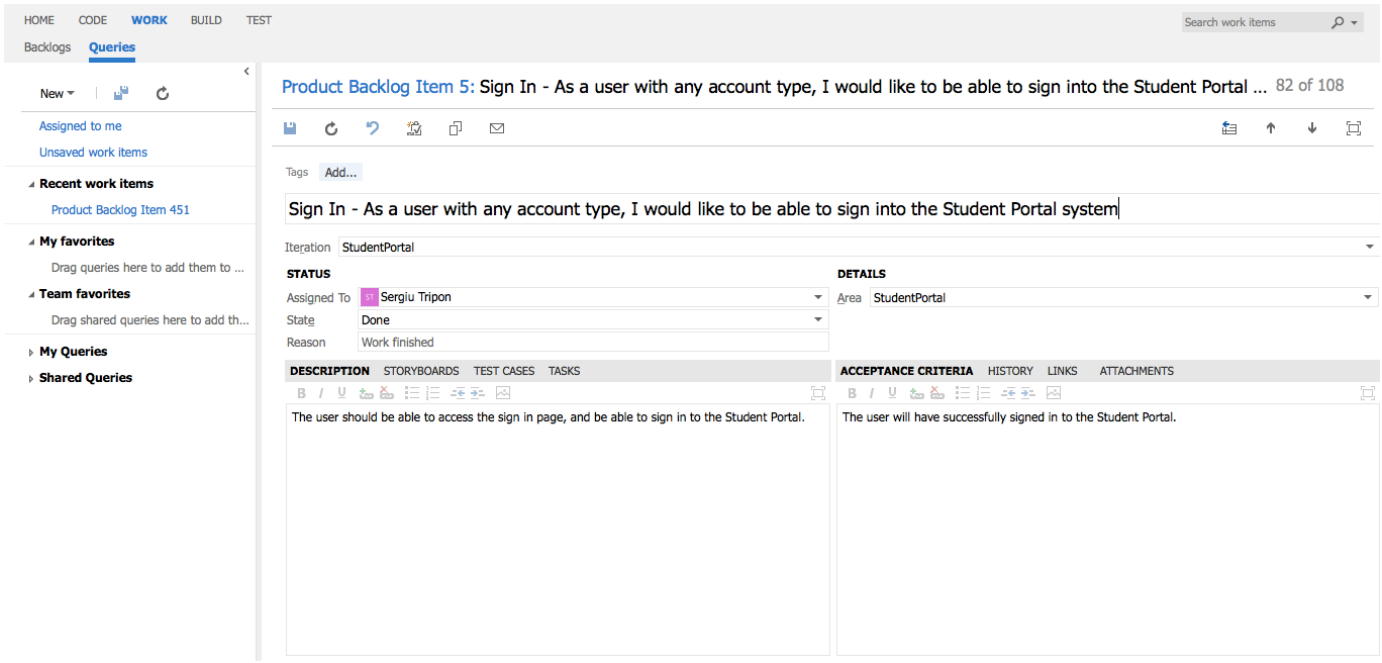
Account - detailed view of requirement



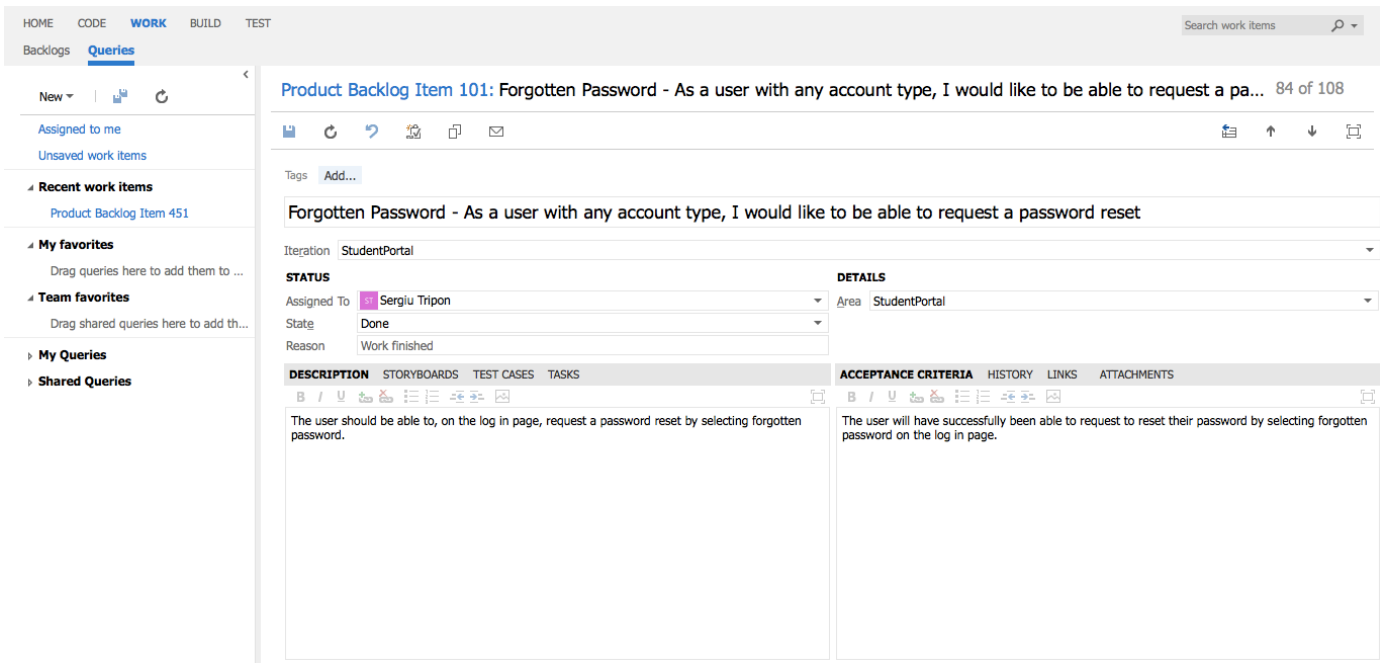
2.2. Common requirements

2.2.1. Student, Academic staff, Administrator

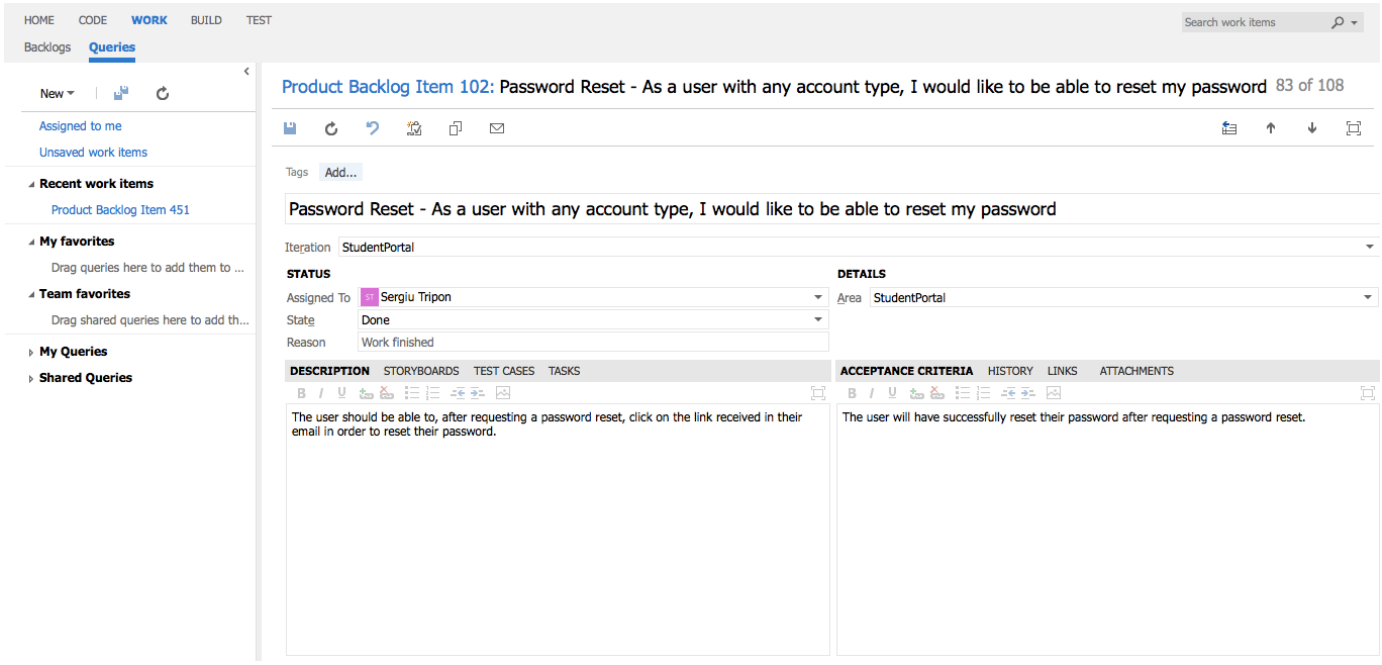
Sign In - detailed view of requirement



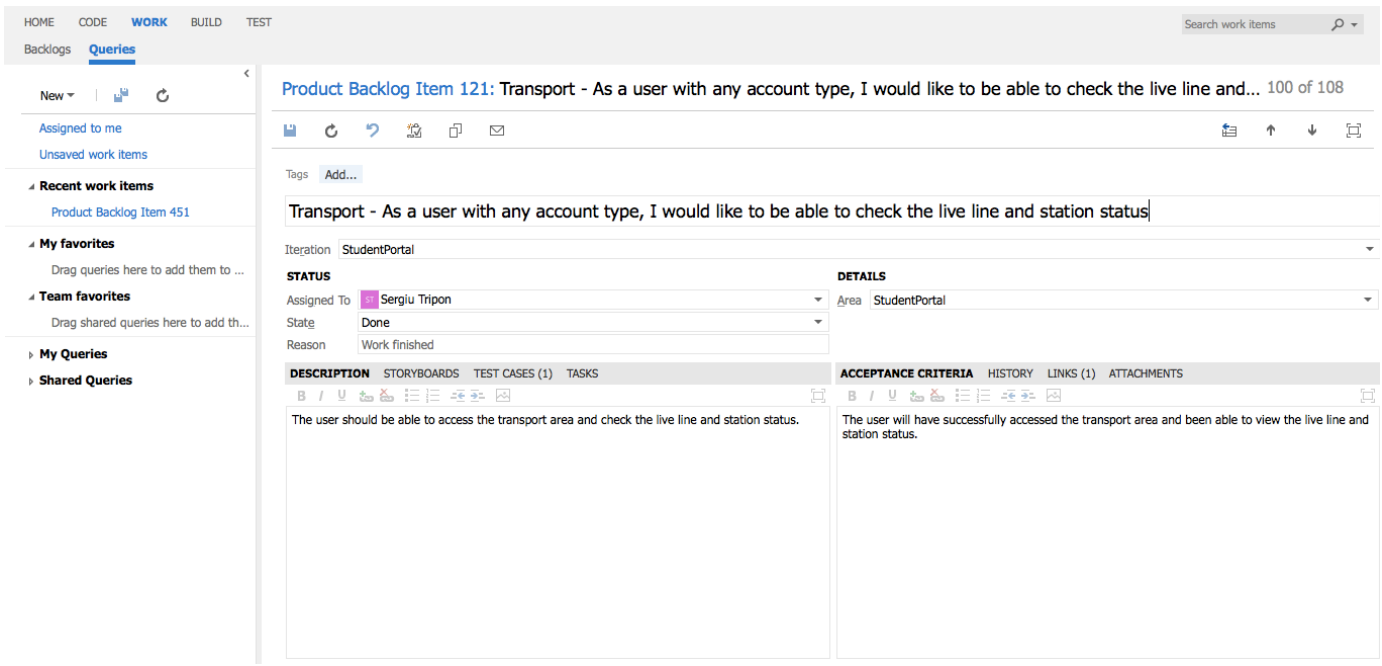
Forgotten Password - detailed view of requirement



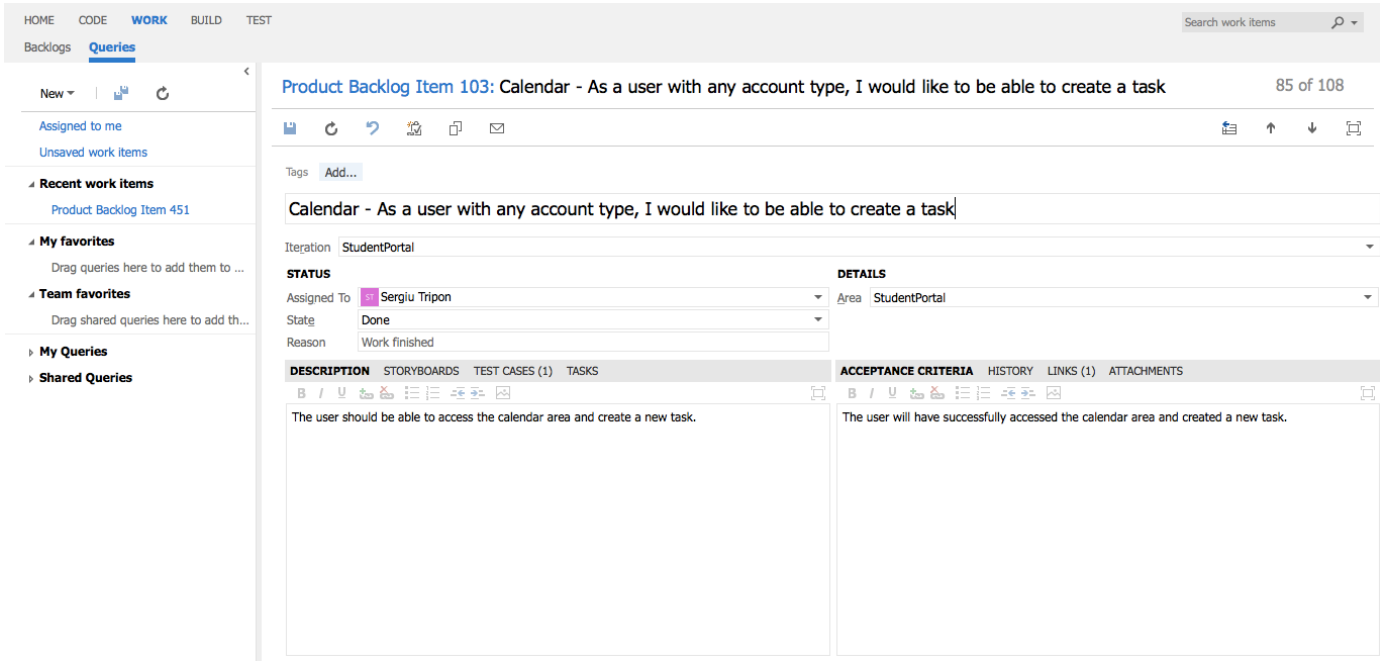
Password Reset - detailed view of requirement



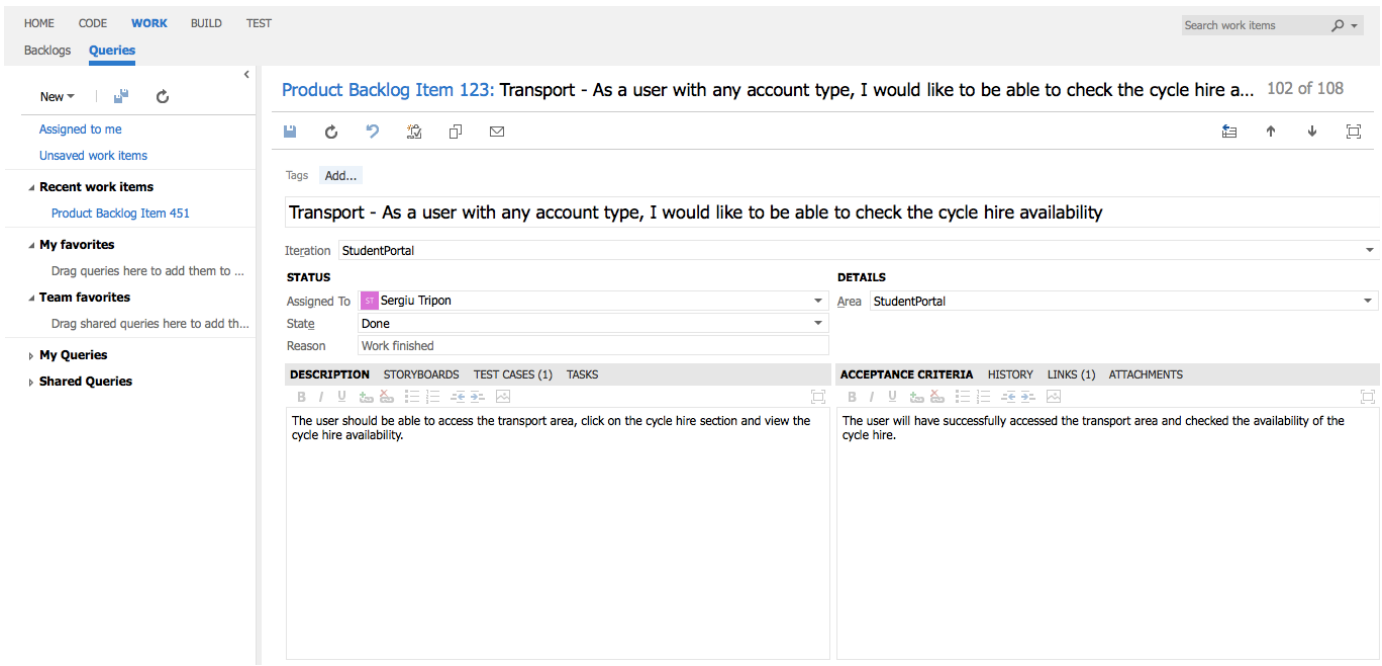
Transport - detailed view of requirement



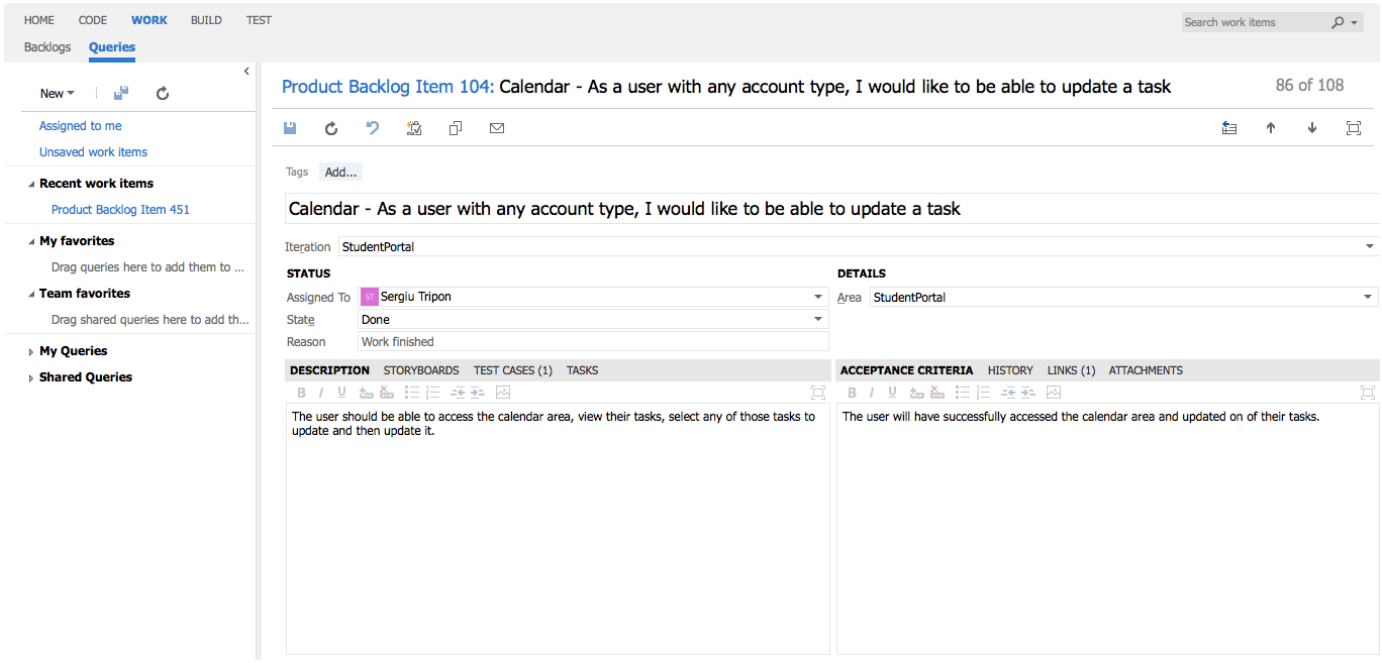
Calendar - detailed view of requirement



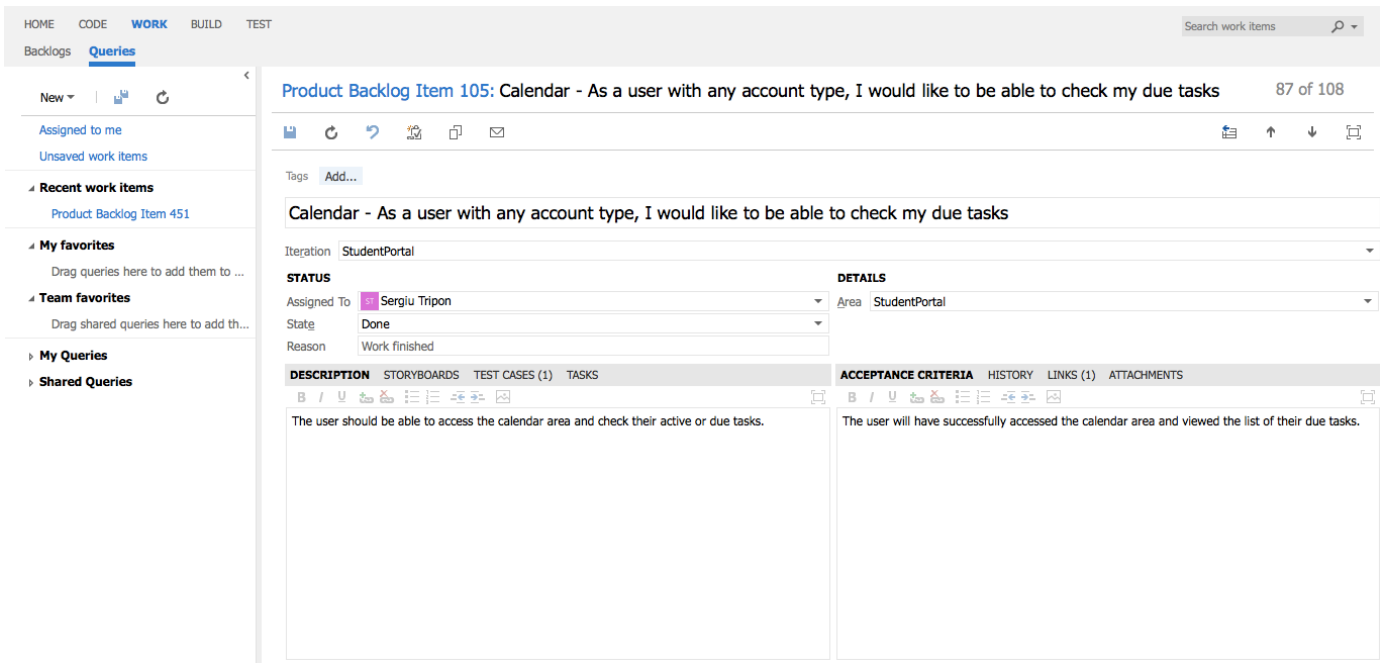
Transport - detailed view of requirement



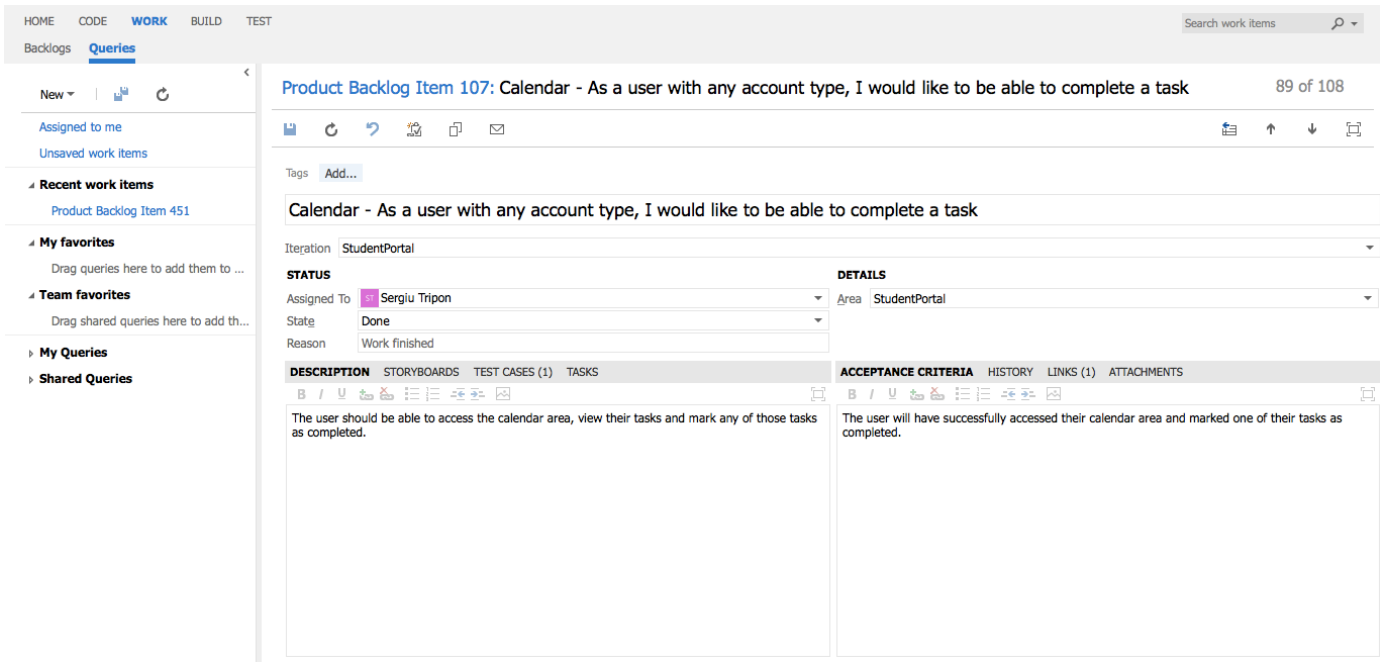
Calendar - detailed view of requirement



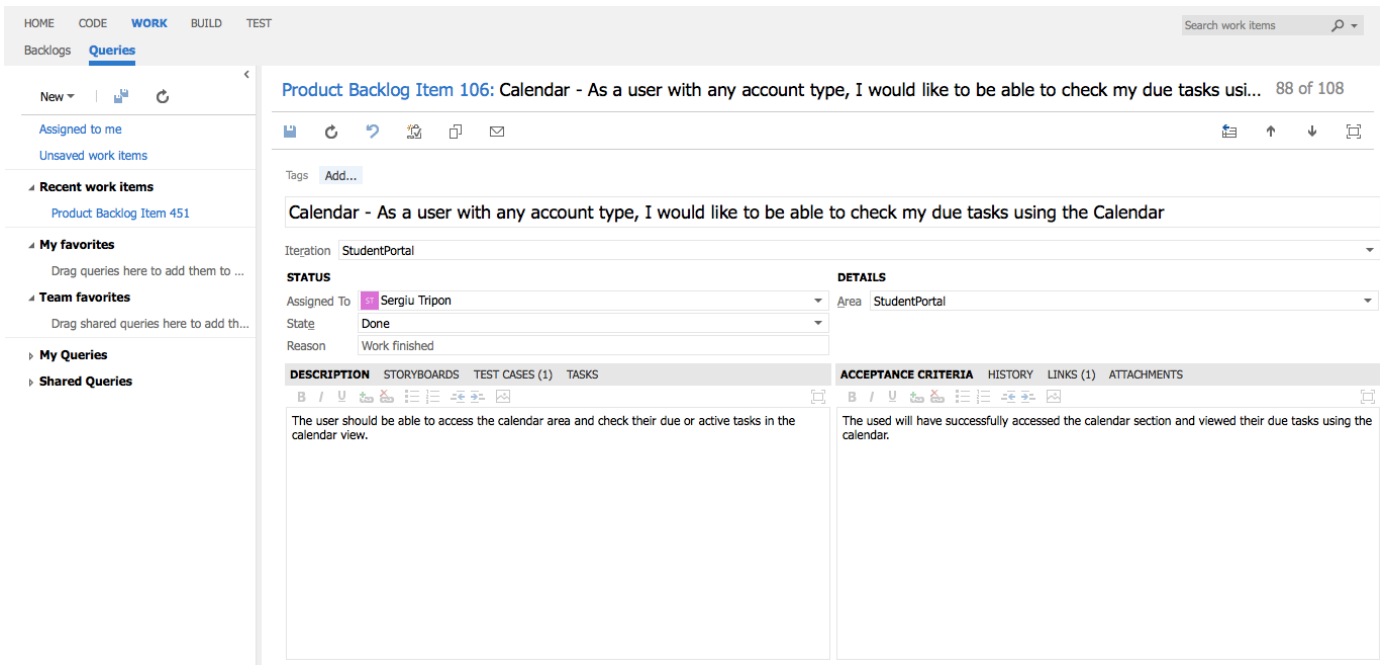
Calendar - detailed view of requirement



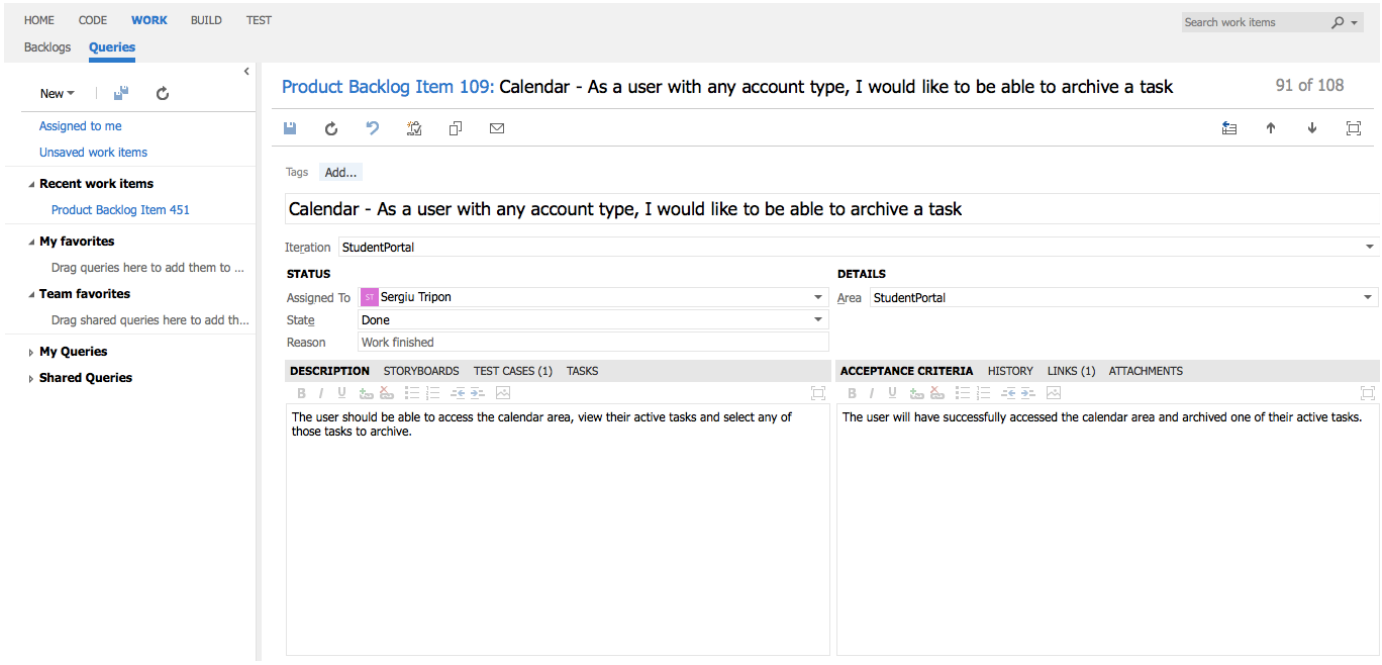
Calendar - detailed view of requirement



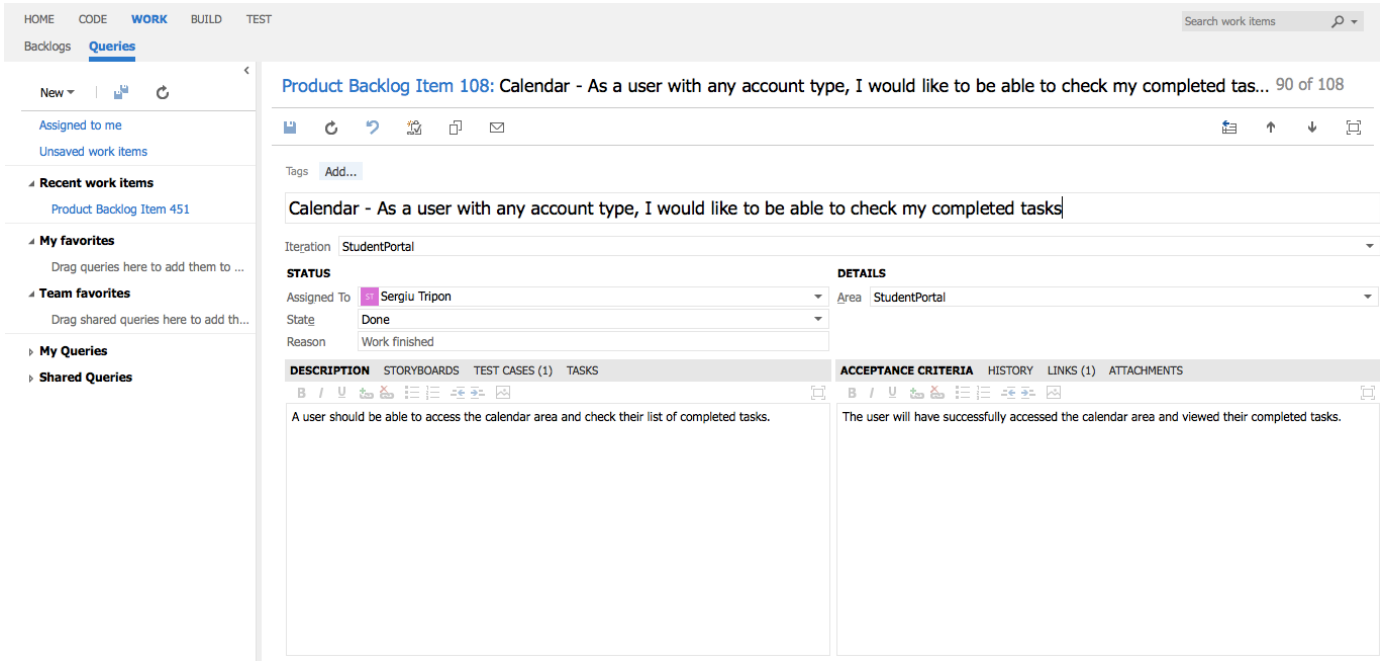
Calendar - detailed view of requirement



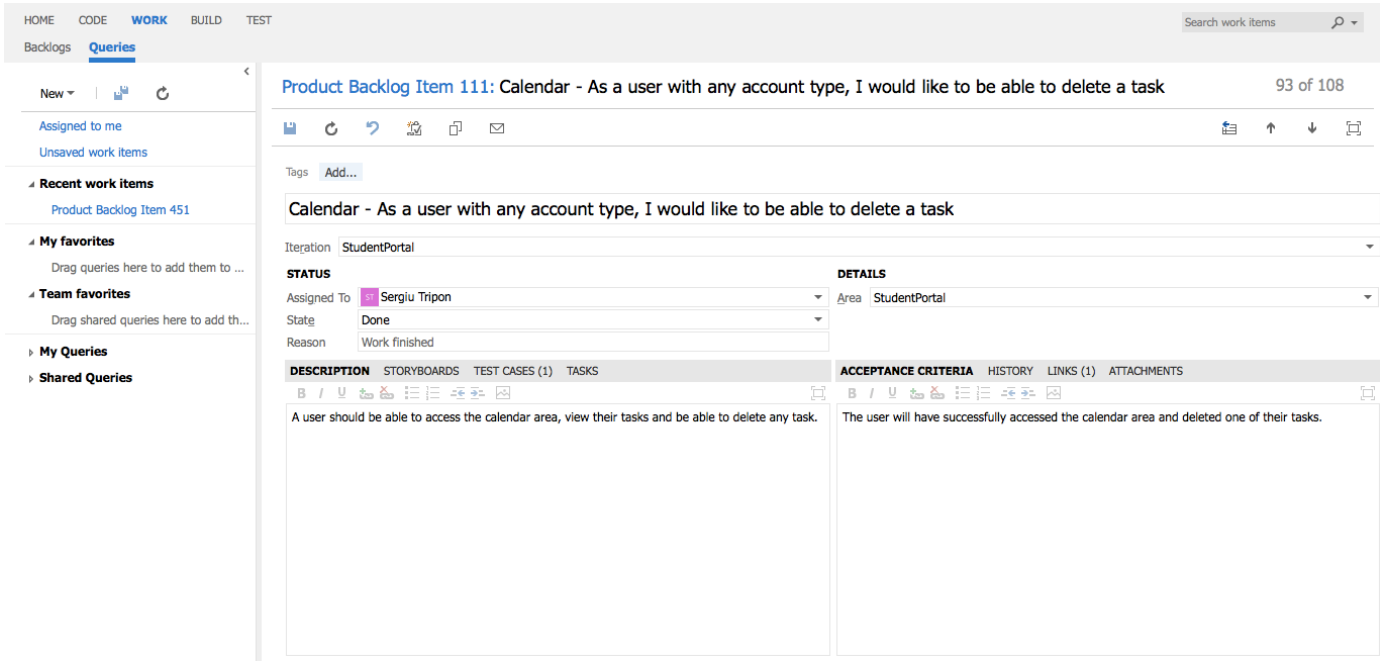
Calendar - detailed view of requirement



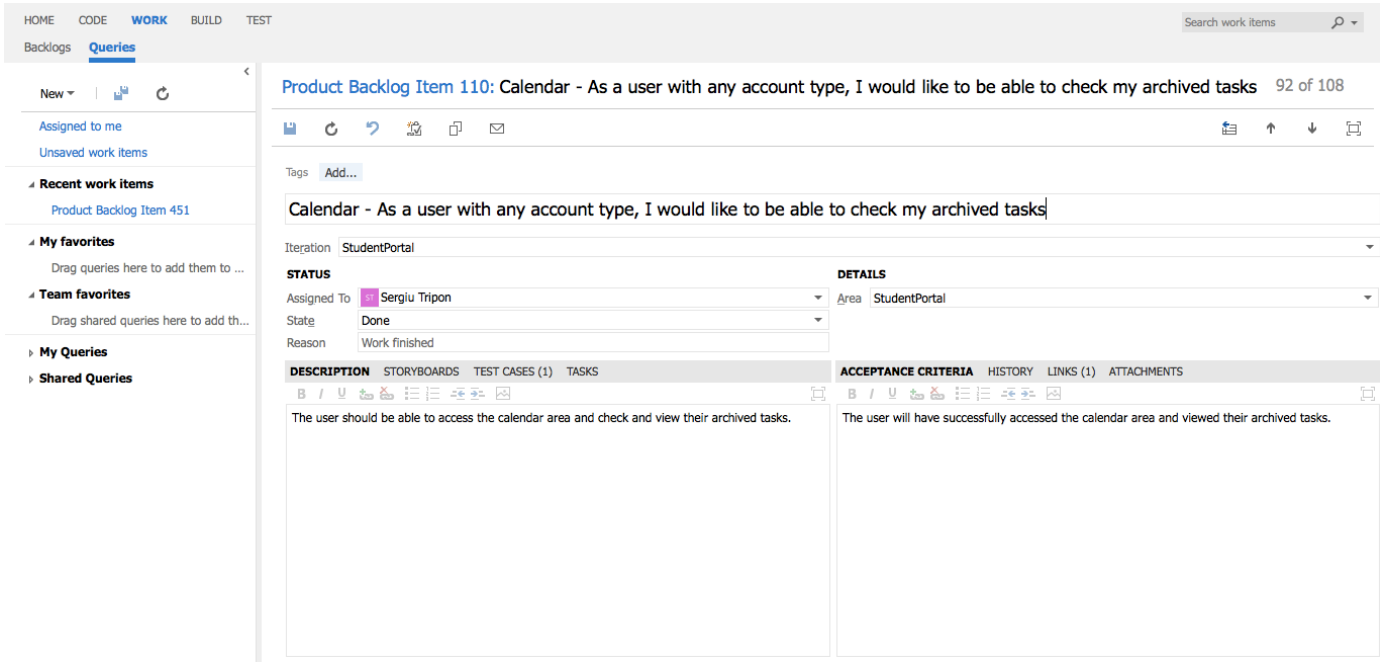
Calendar - detailed view of requirement



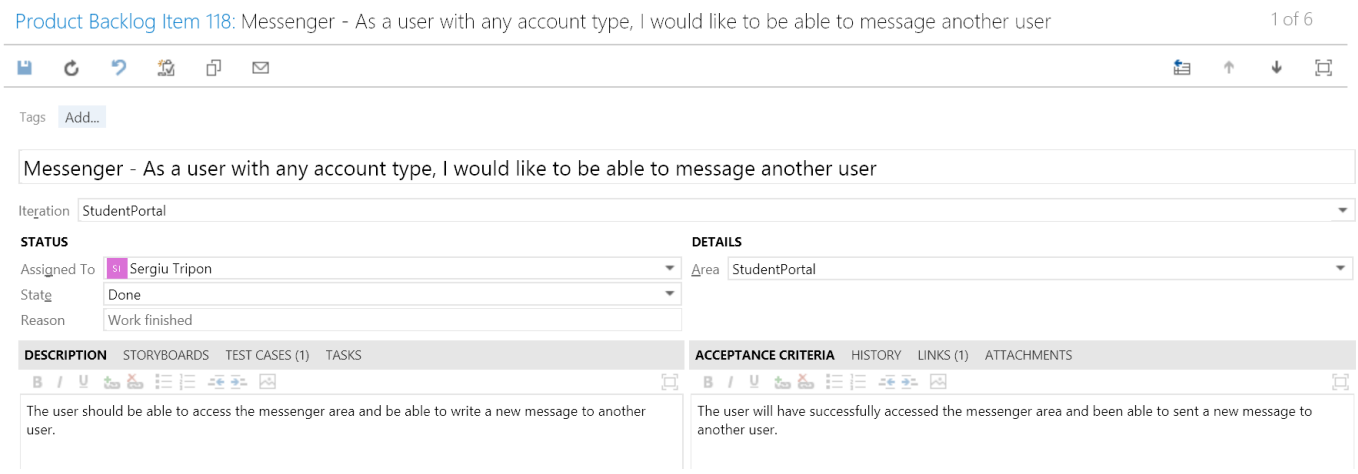
Calendar - detailed view of requirement



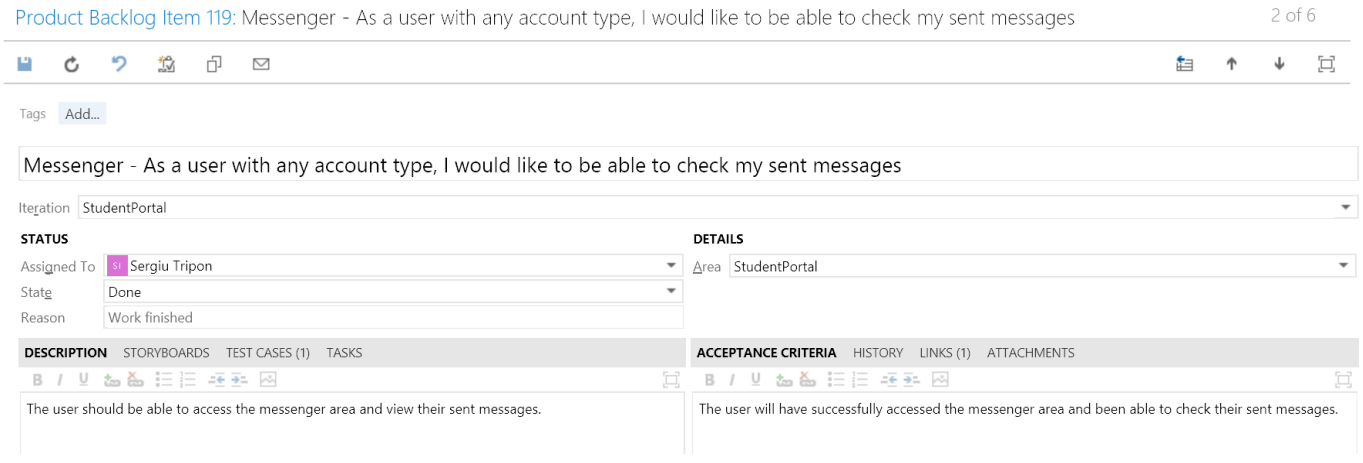
Calendar - detailed view of requirement



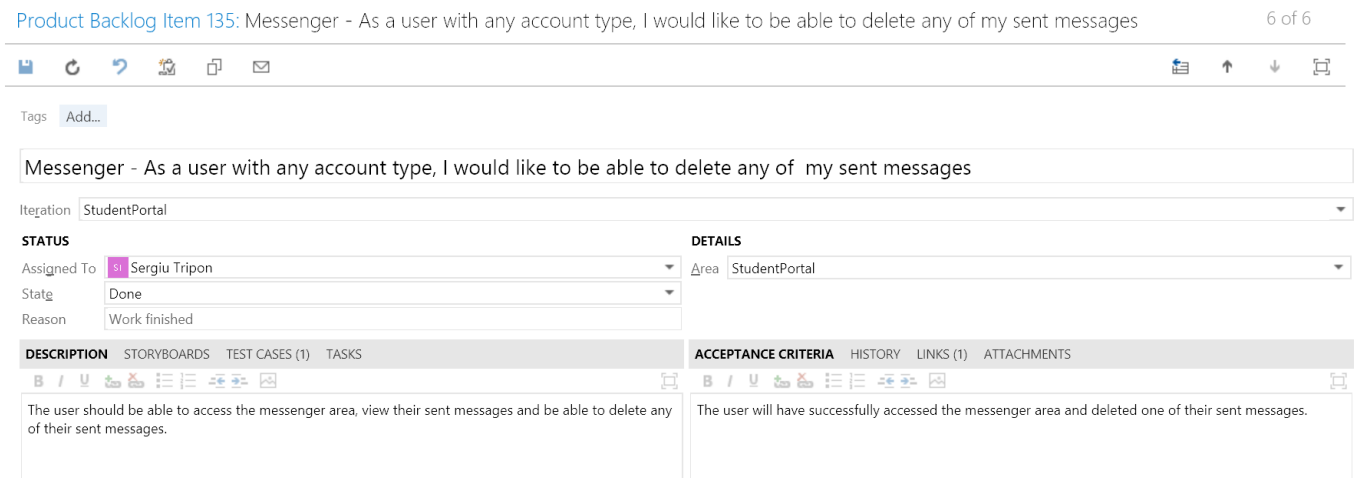
Messenger - detailed view of requirement



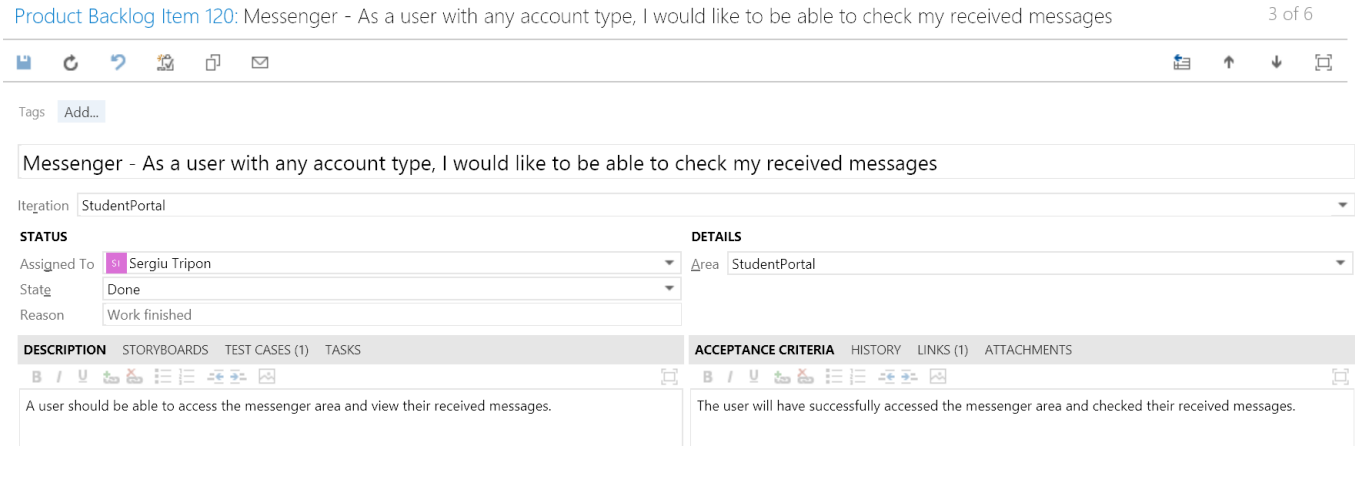
Messenger - detailed view of requirement



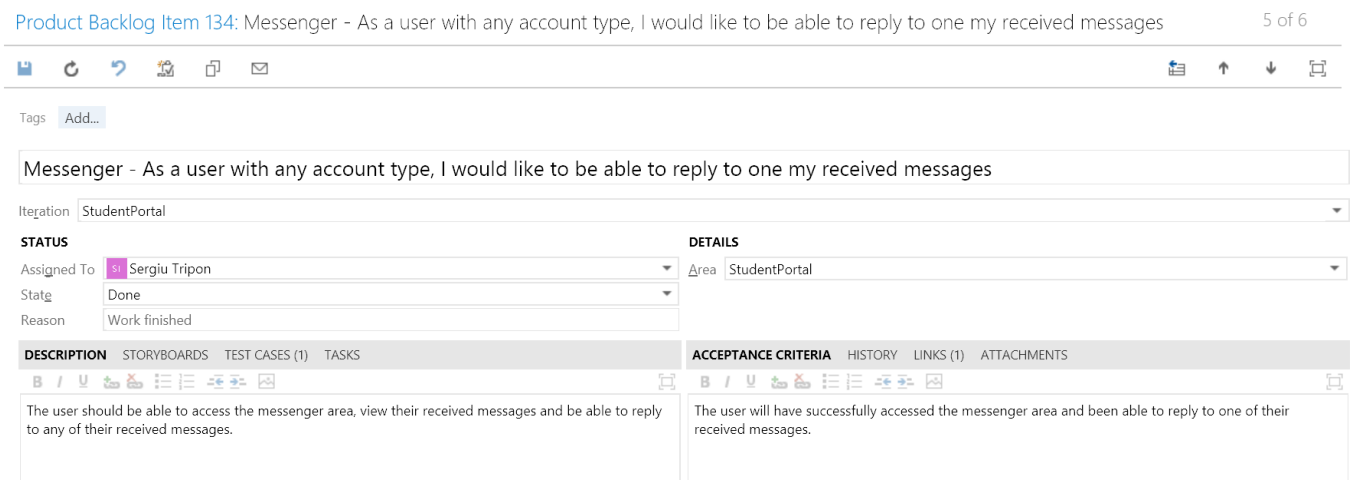
Messenger - detailed view of requirement



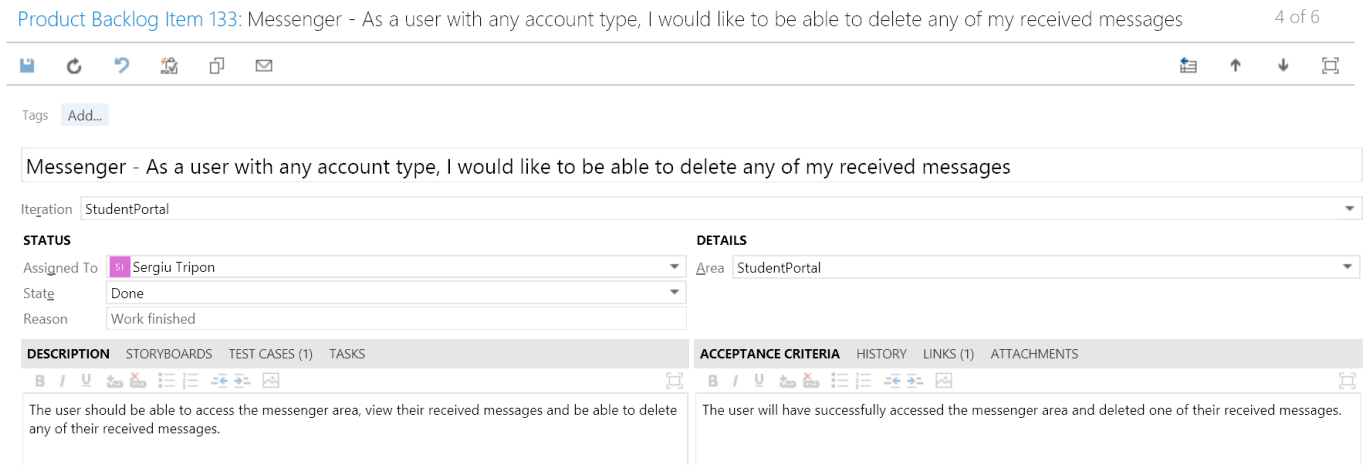
Messenger - detailed view of requirement



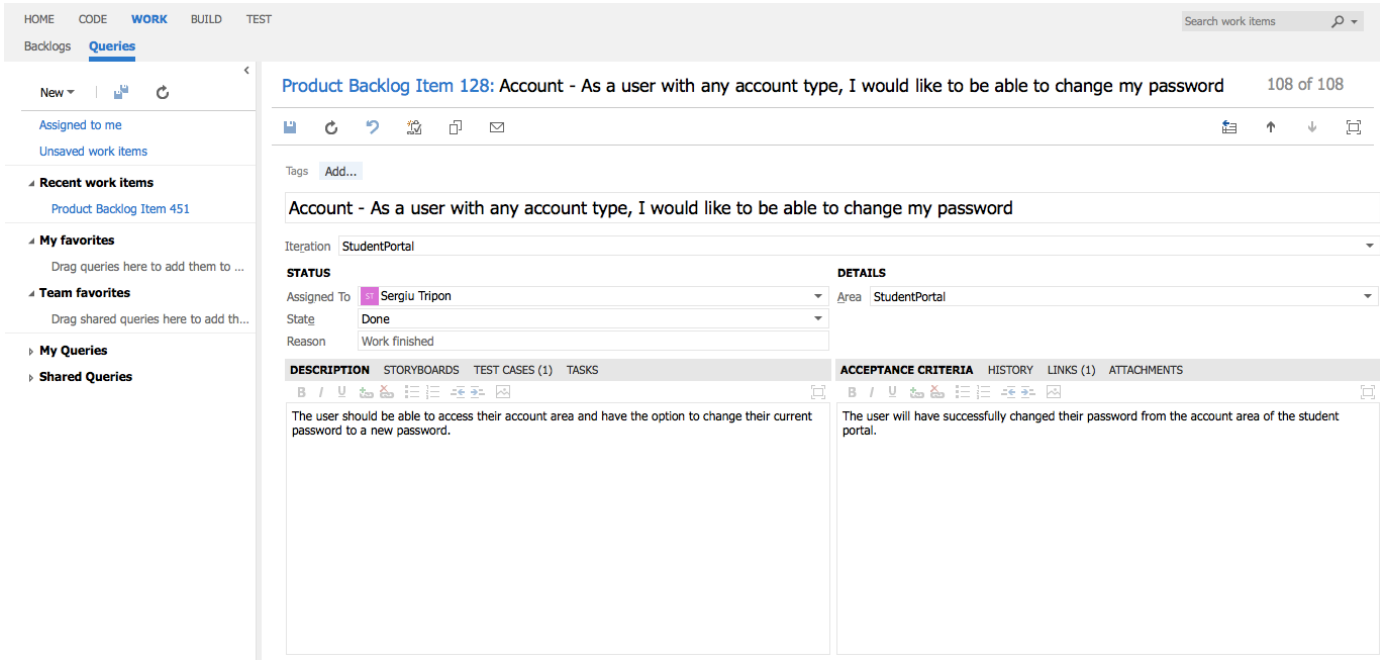
Messenger - detailed view of requirement



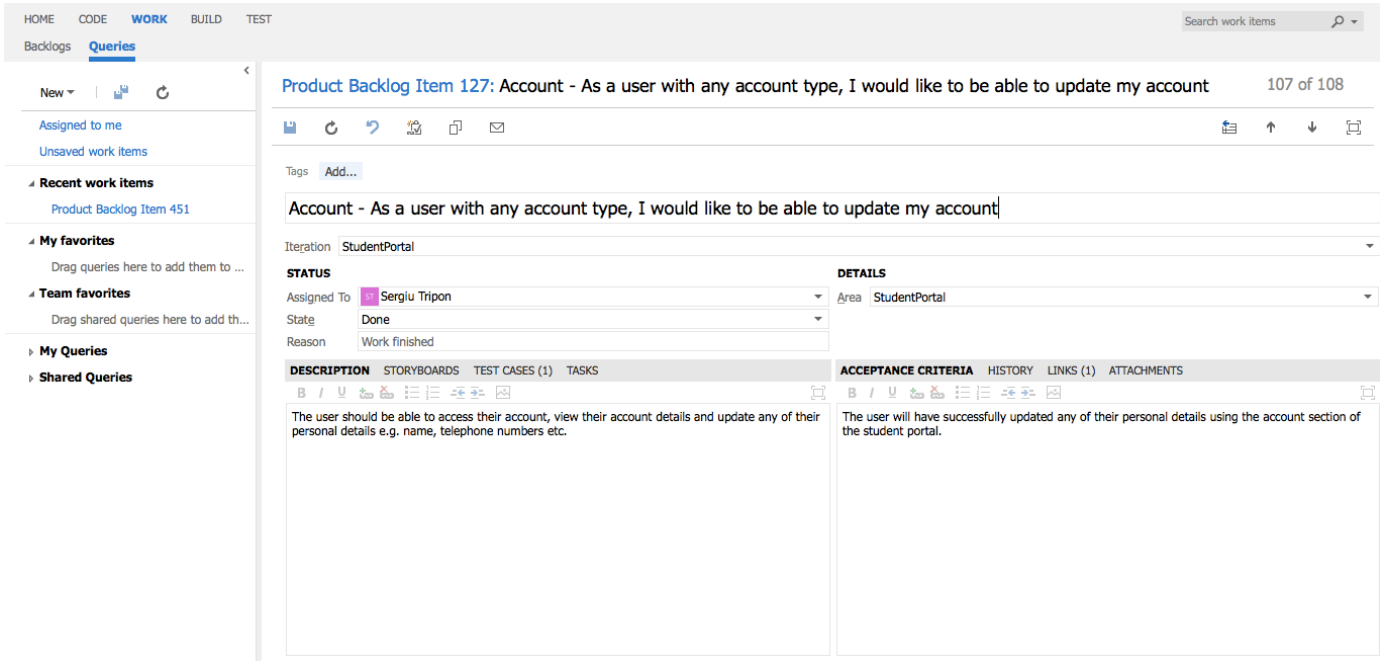
Messenger - detailed view of requirement



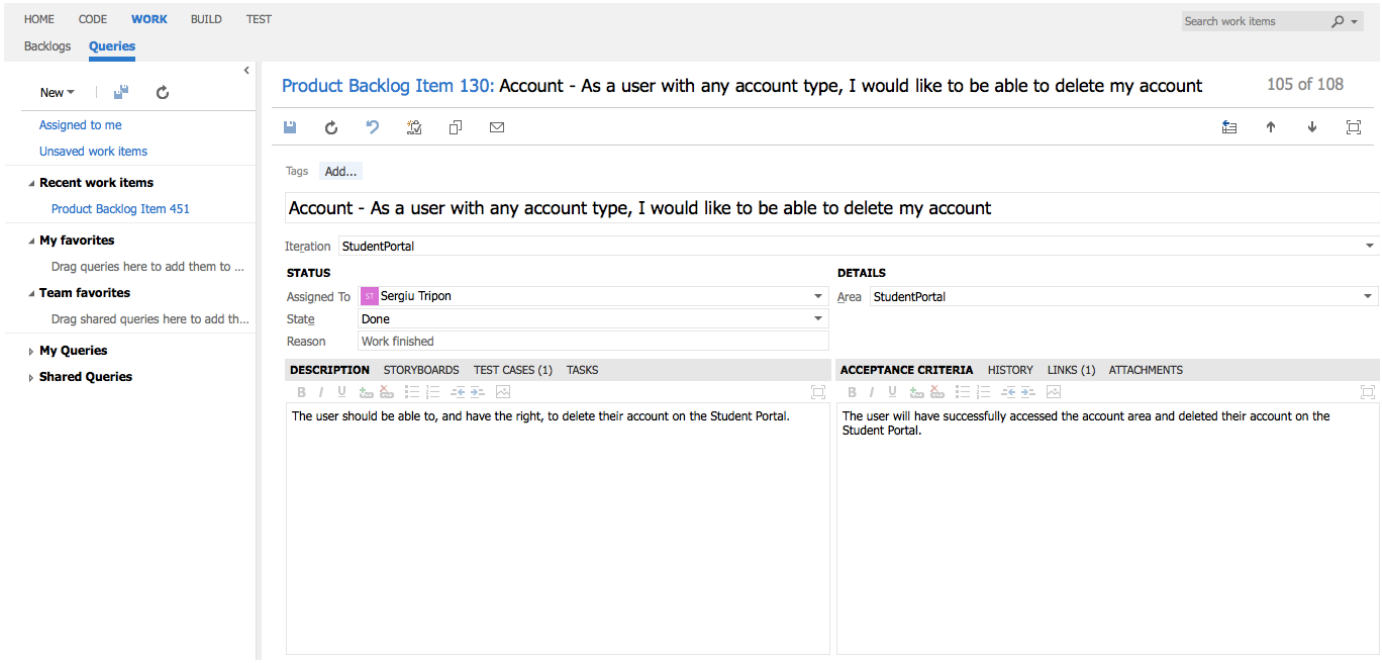
Account - detailed view of requirement



Account - detailed view of requirement

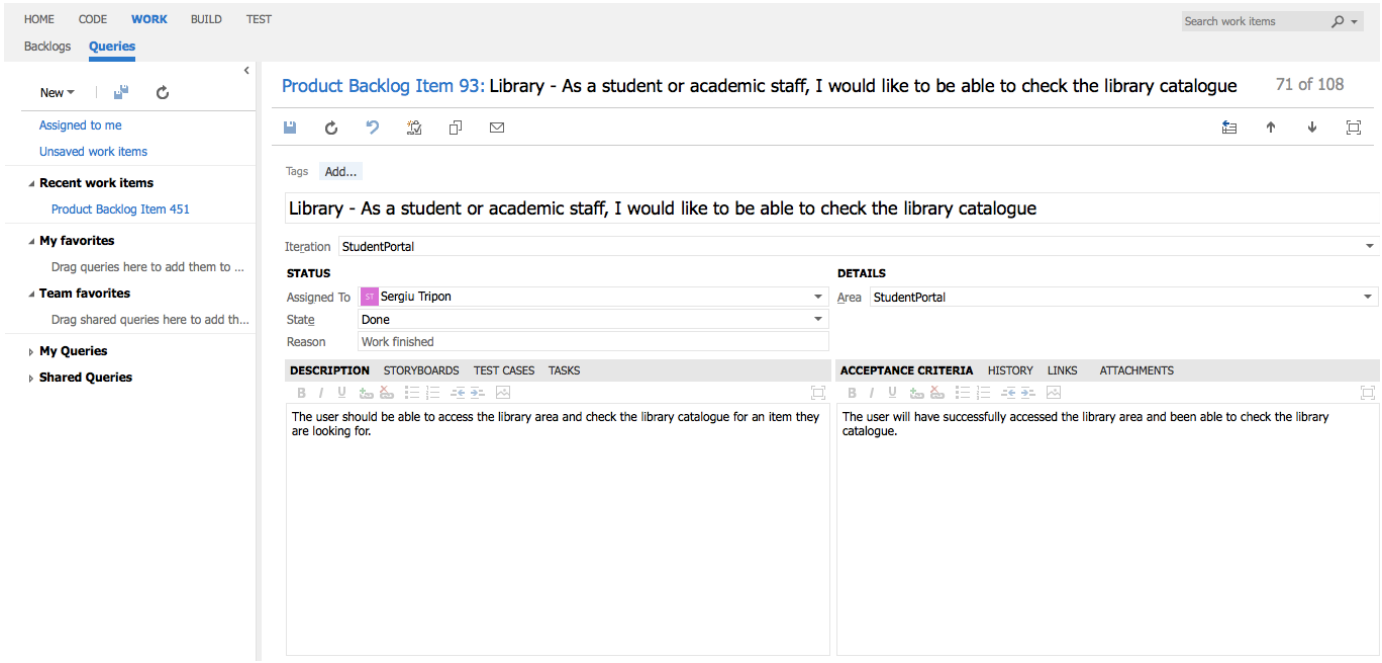


Account - detailed view of requirement

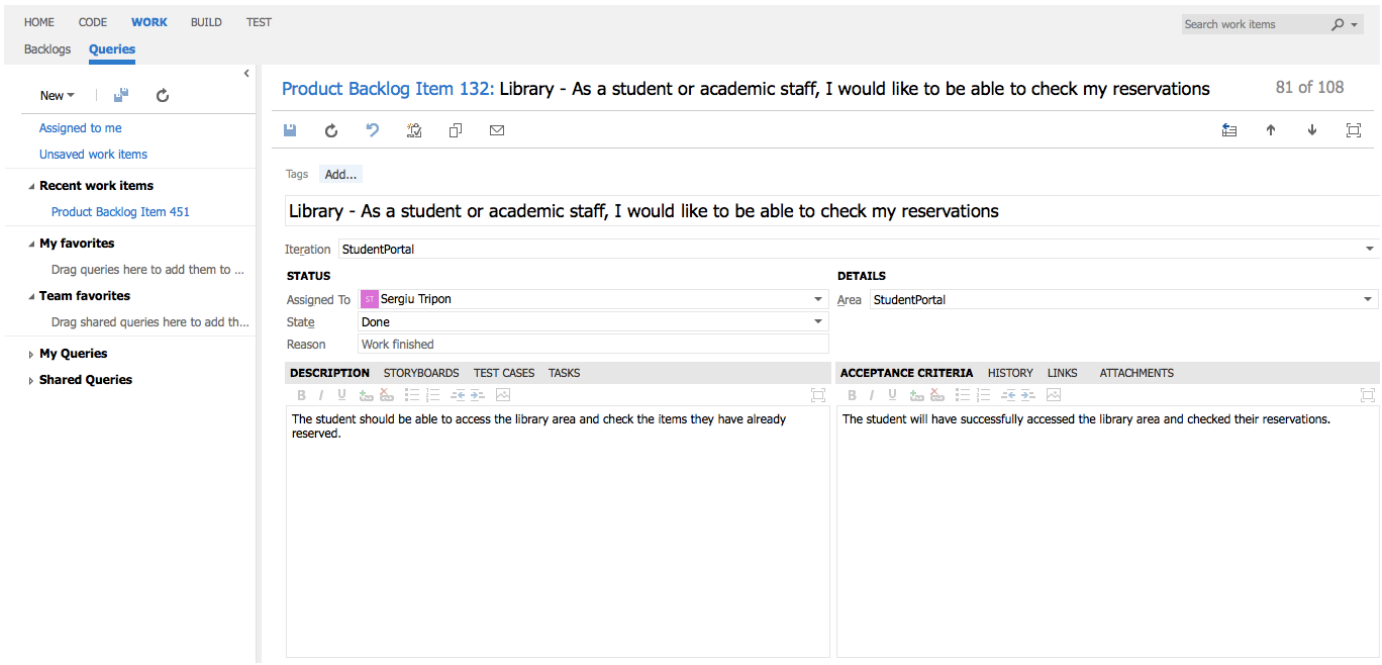


2.2.2. Student, Academic staff

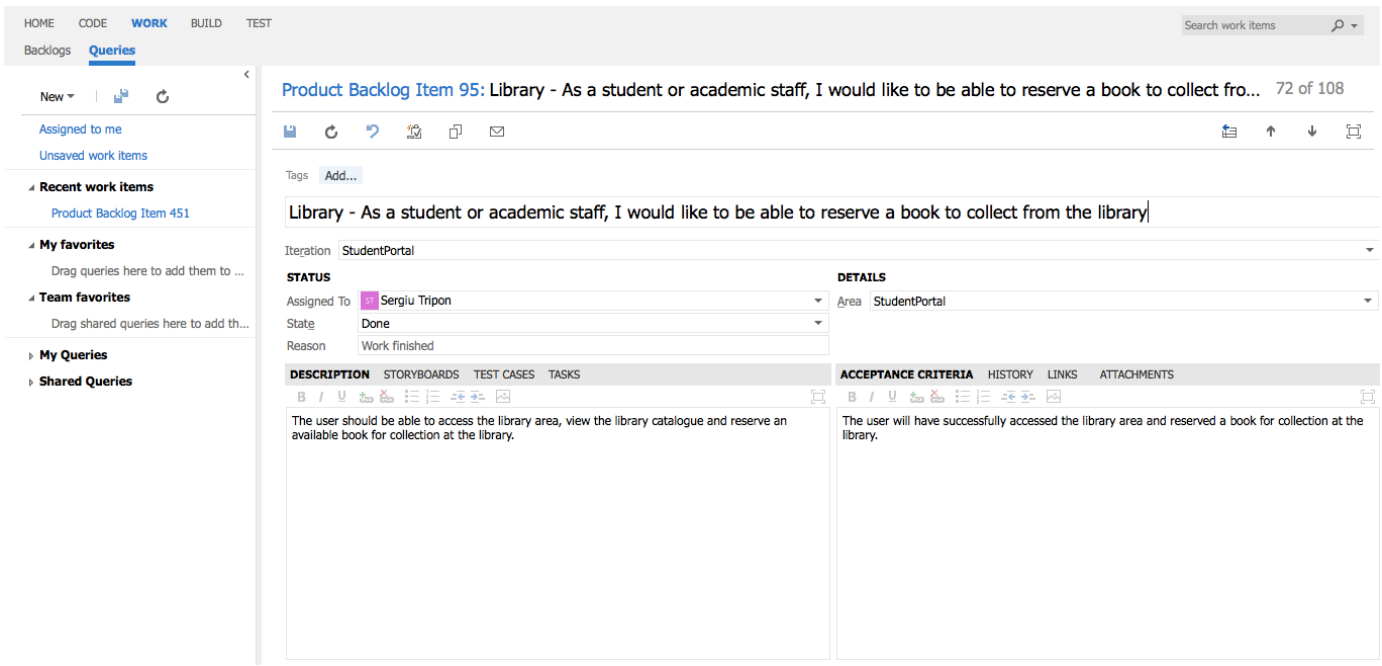
Library - detailed view of requirement



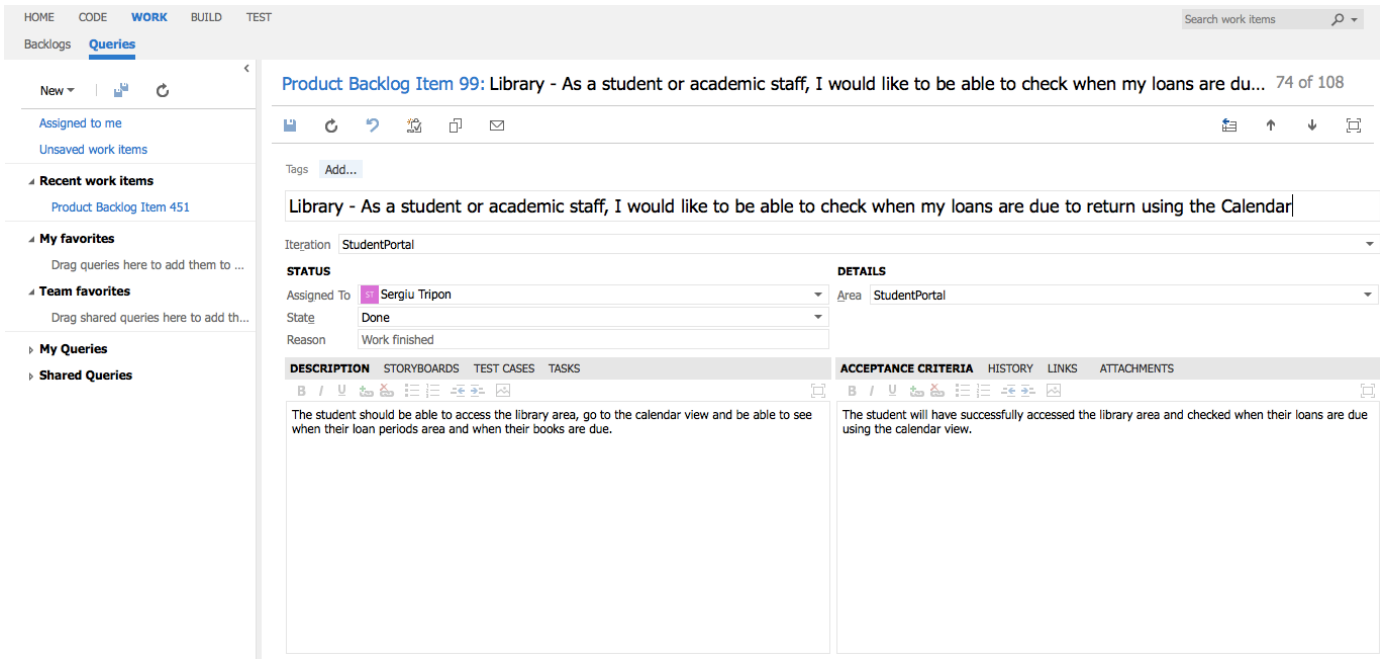
Library - detailed view of requirement



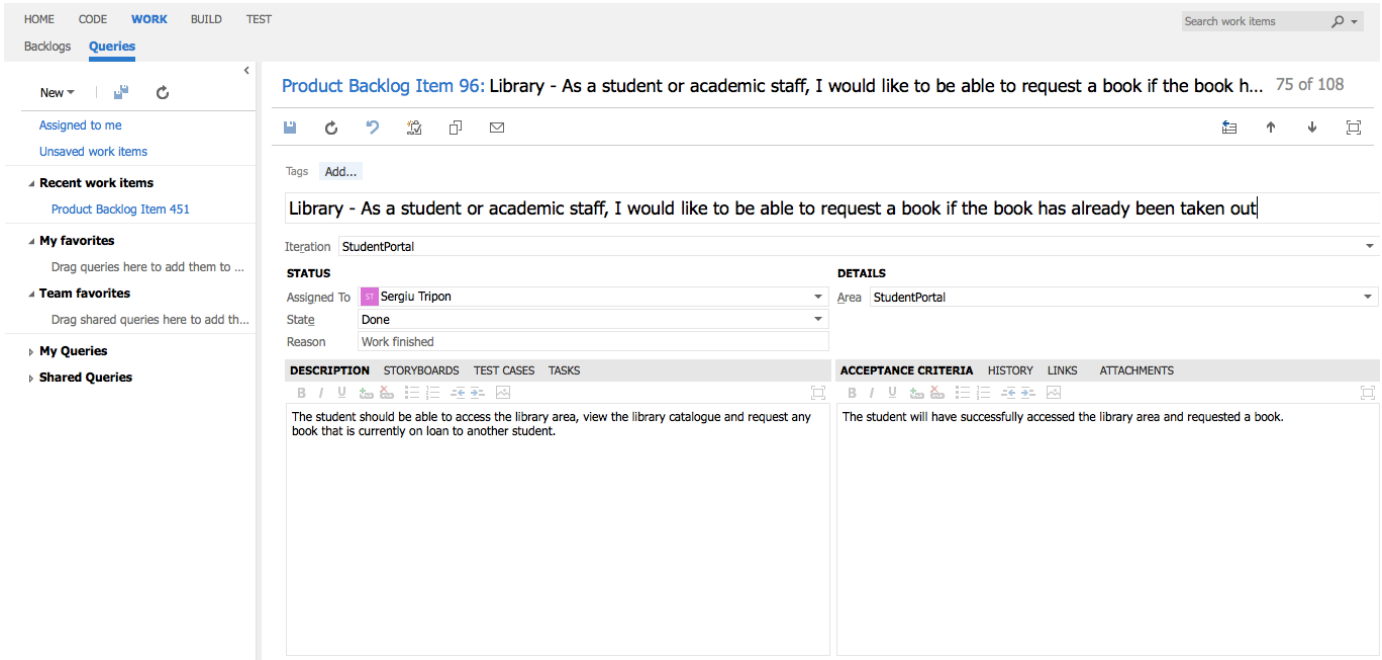
Library - detailed view of requirement



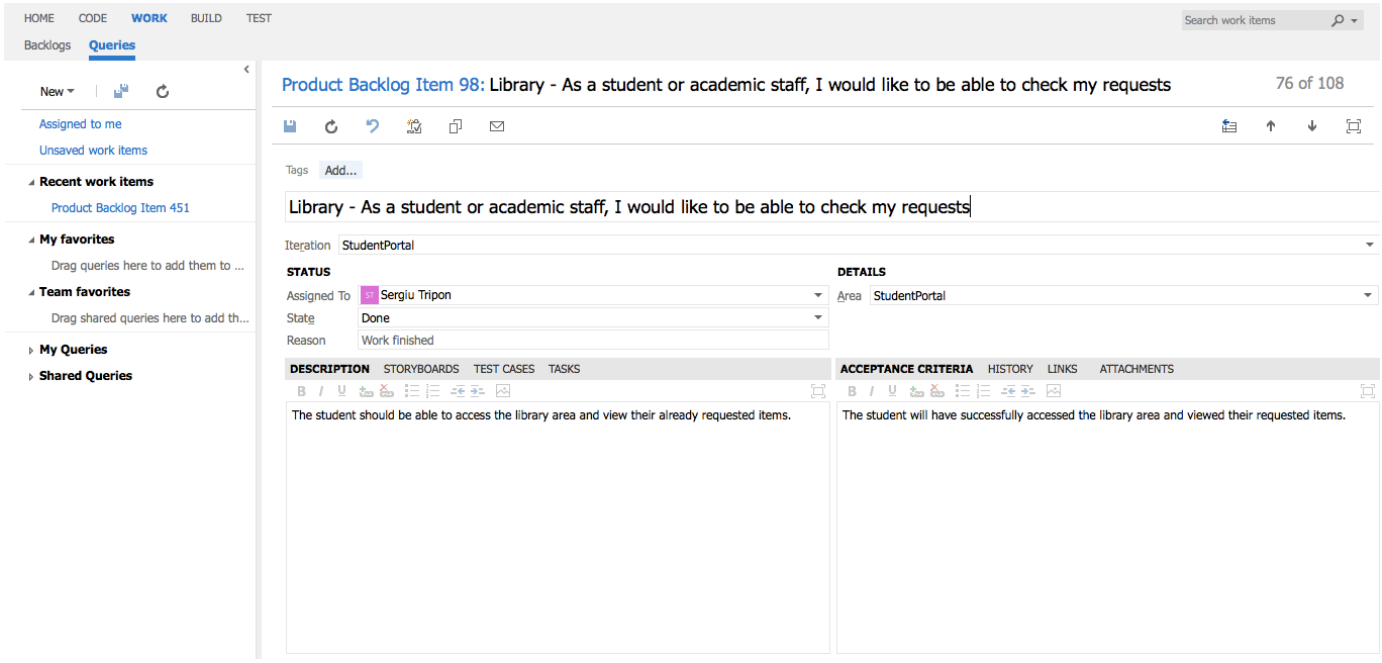
Library - detailed view of requirement



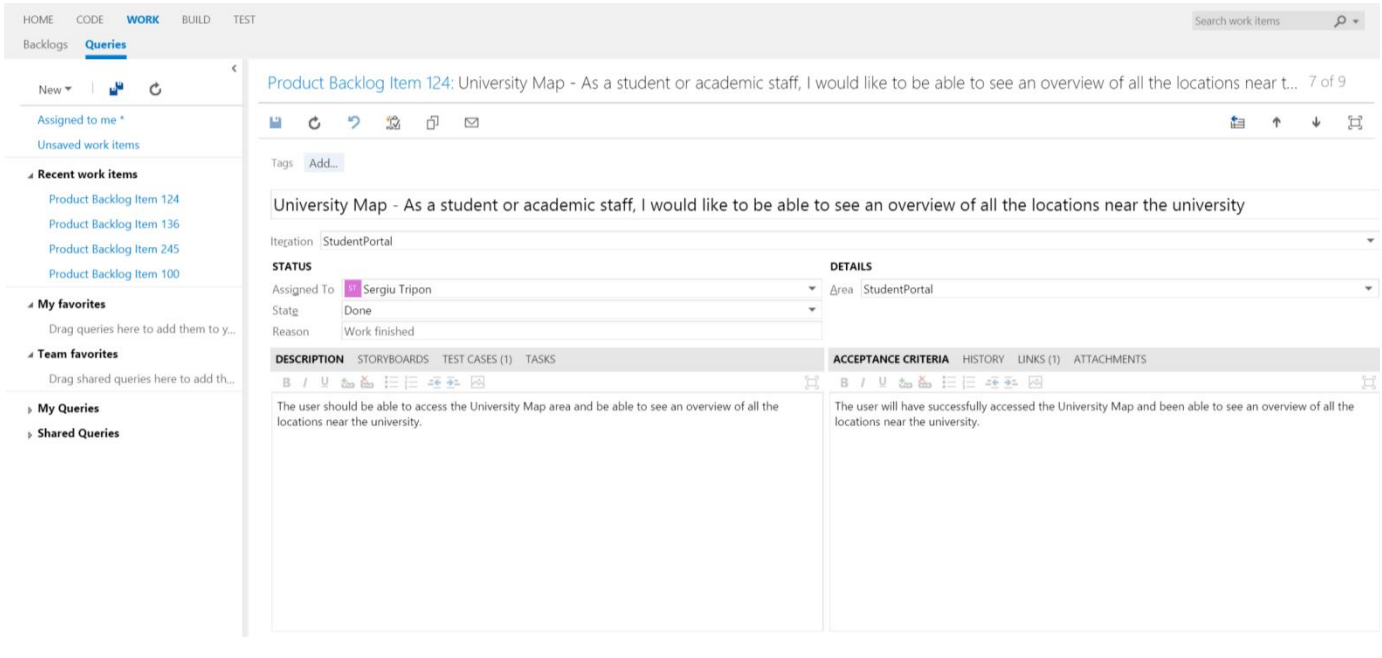
Library - detailed view of requirement



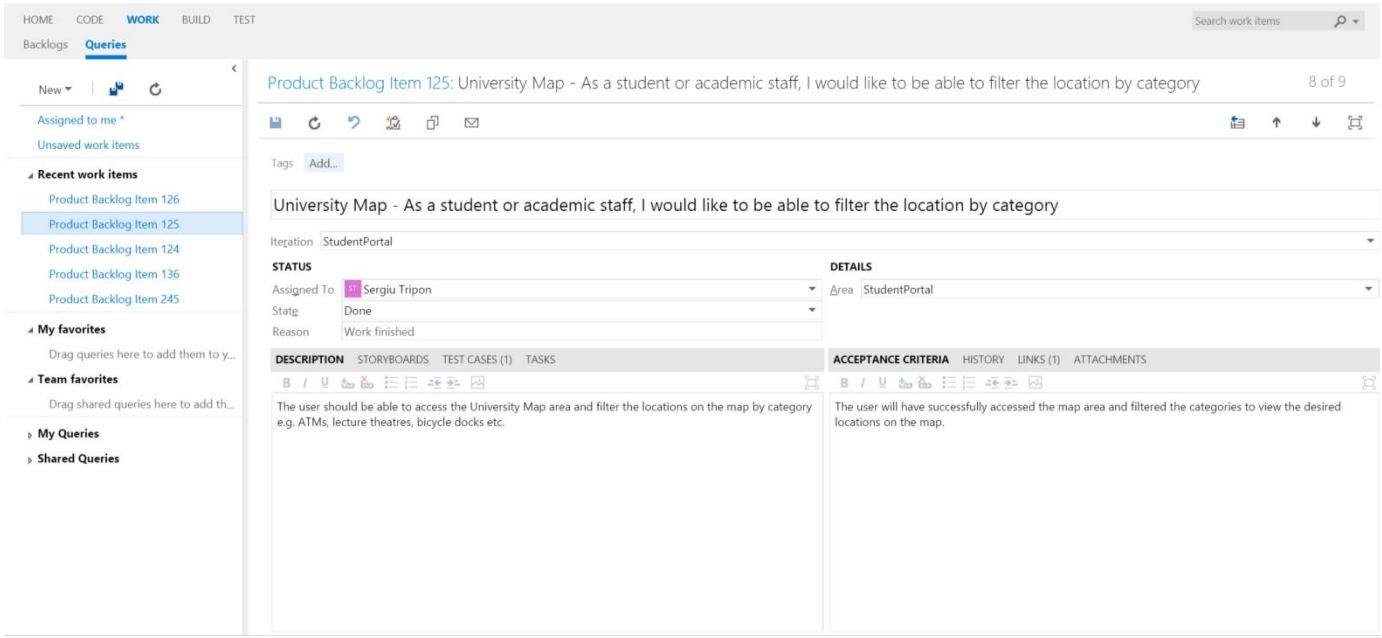
Library - detailed view of requirement



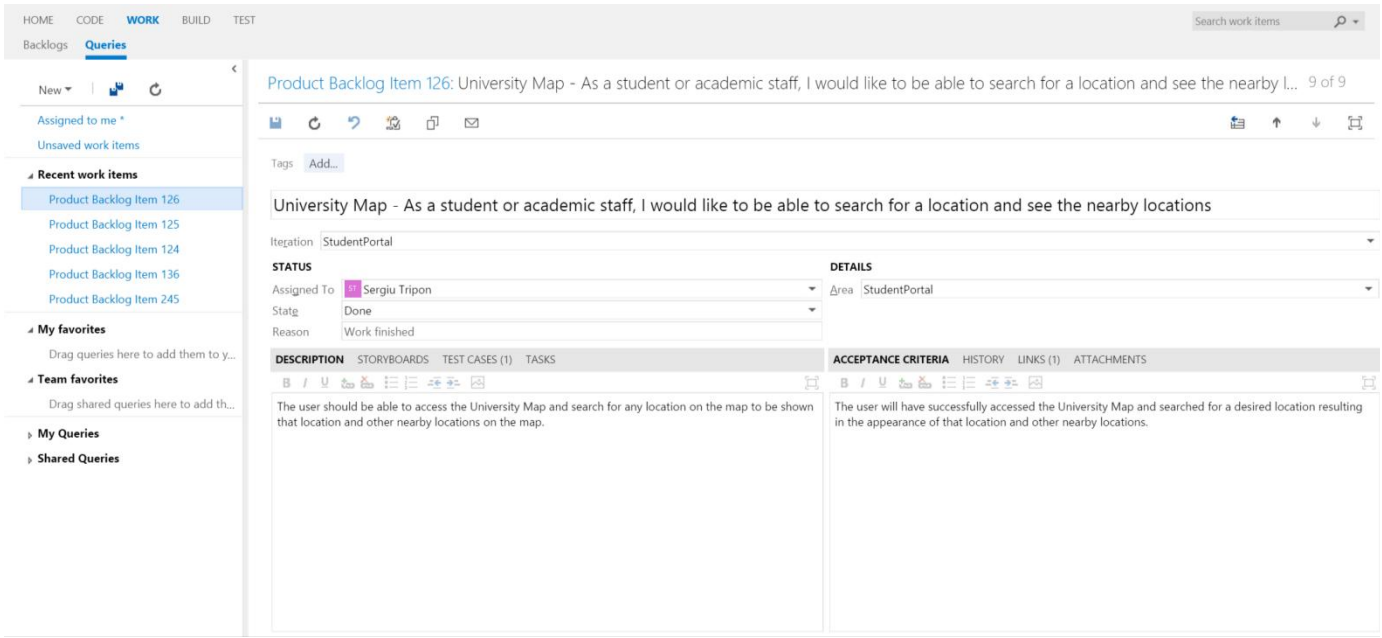
University Map - detailed view of requirement



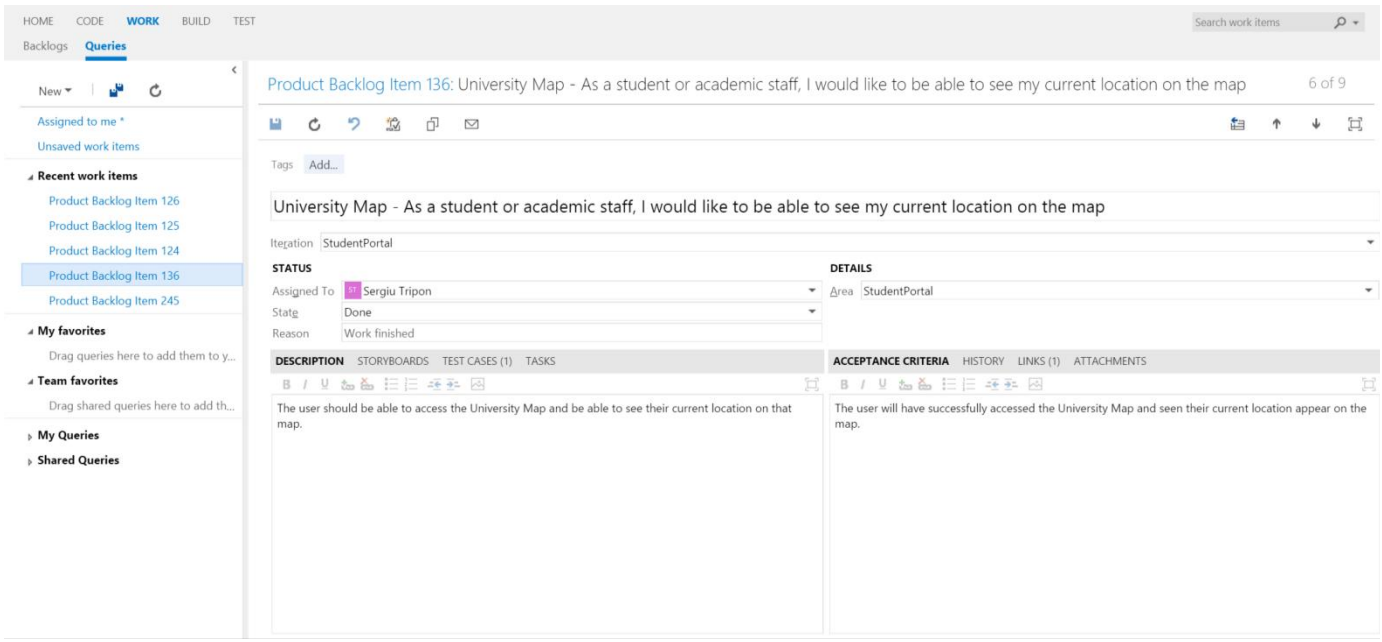
University Map - detailed view of requirement



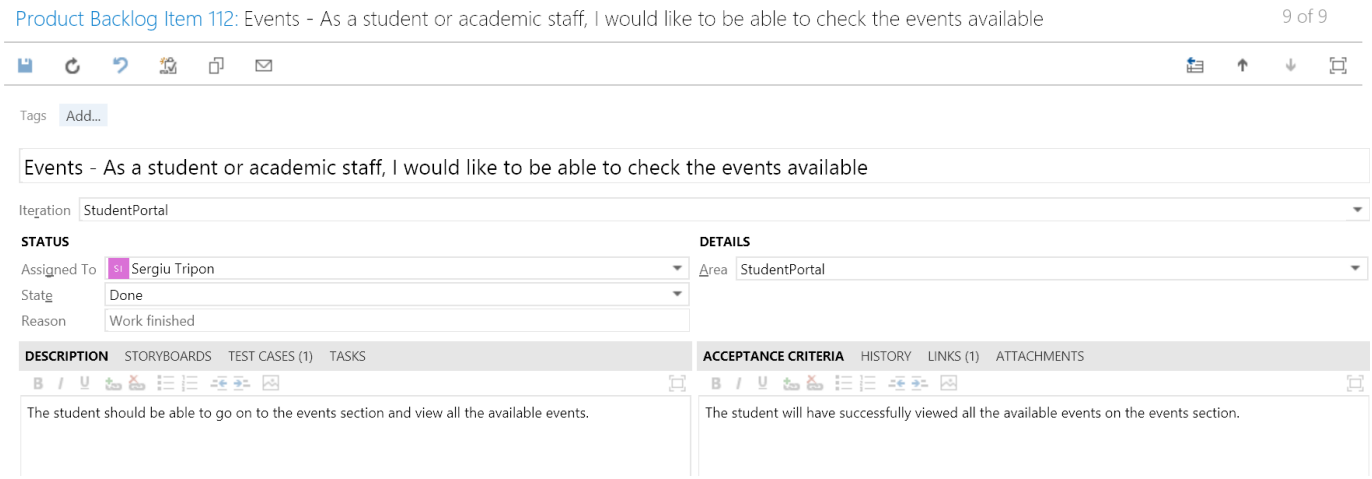
University Map - detailed view of requirement



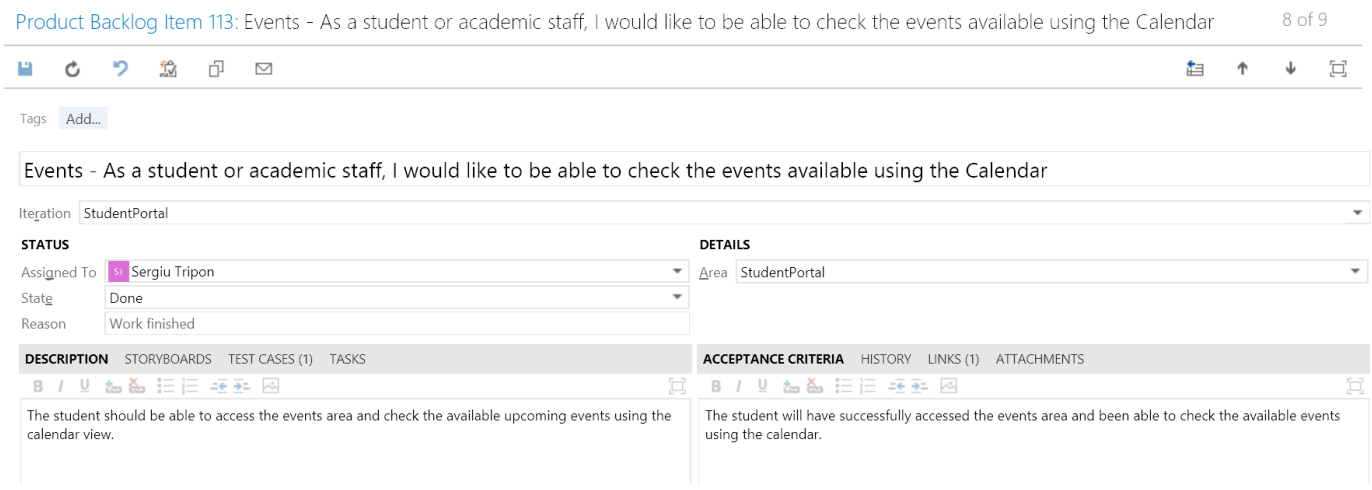
University Map - detailed view of requirement



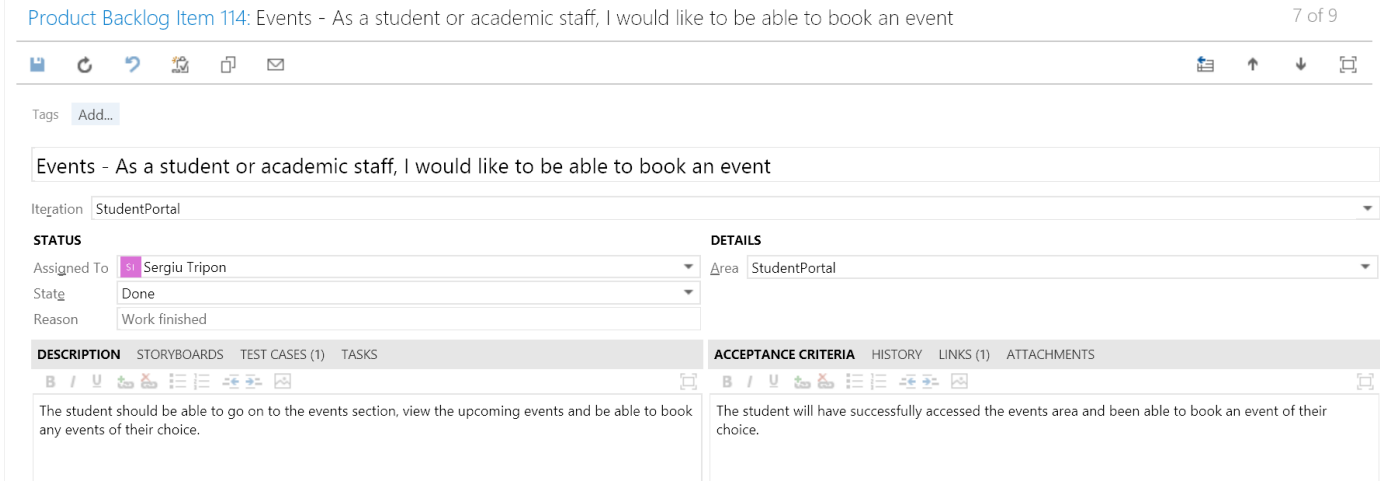
Events - detailed view of requirement



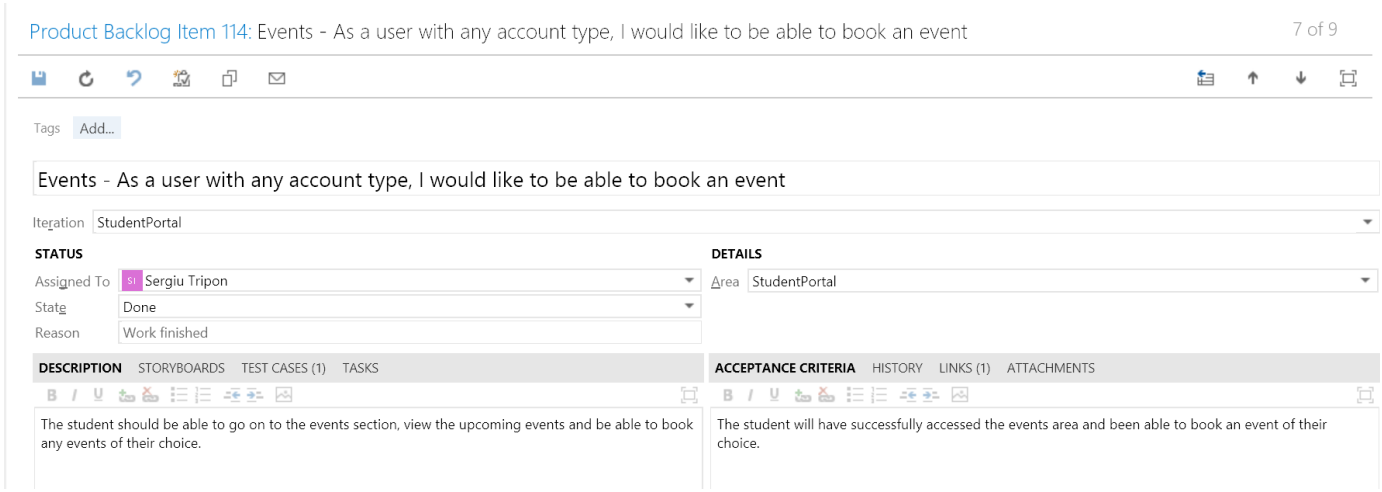
Events - detailed view of requirement



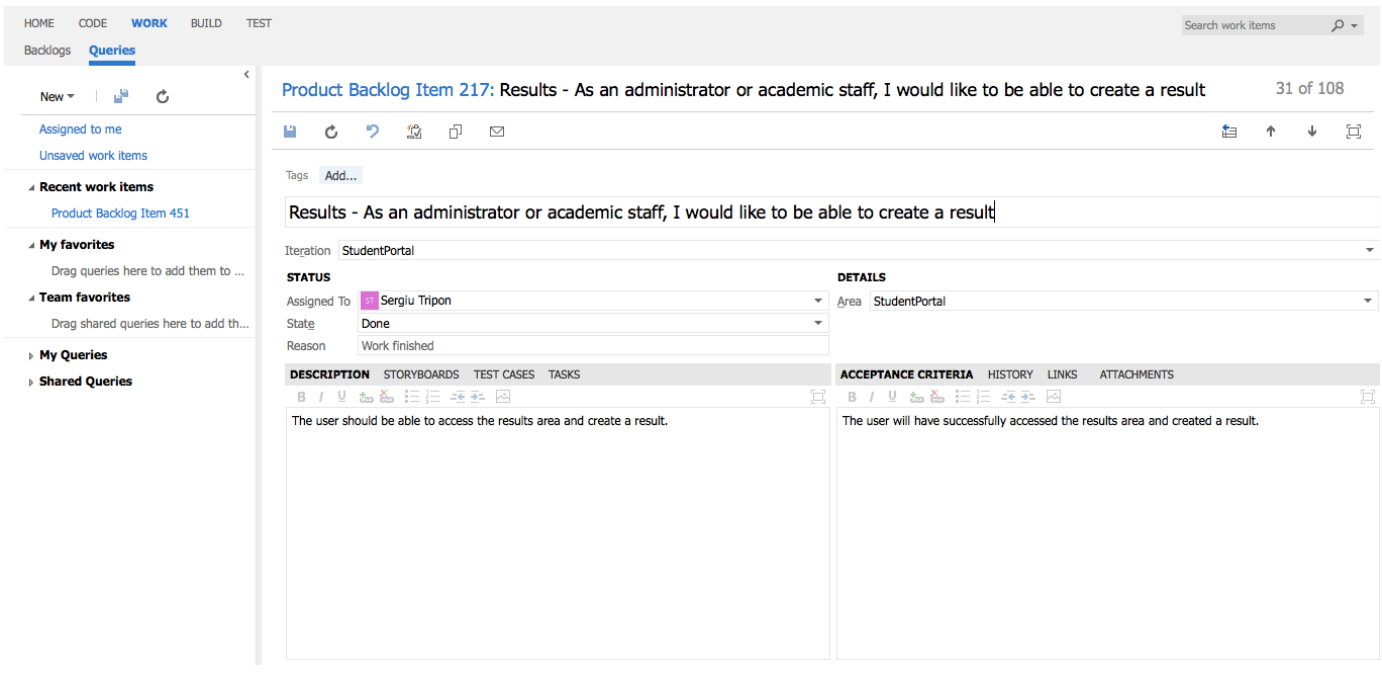
Events - detailed view of requirement



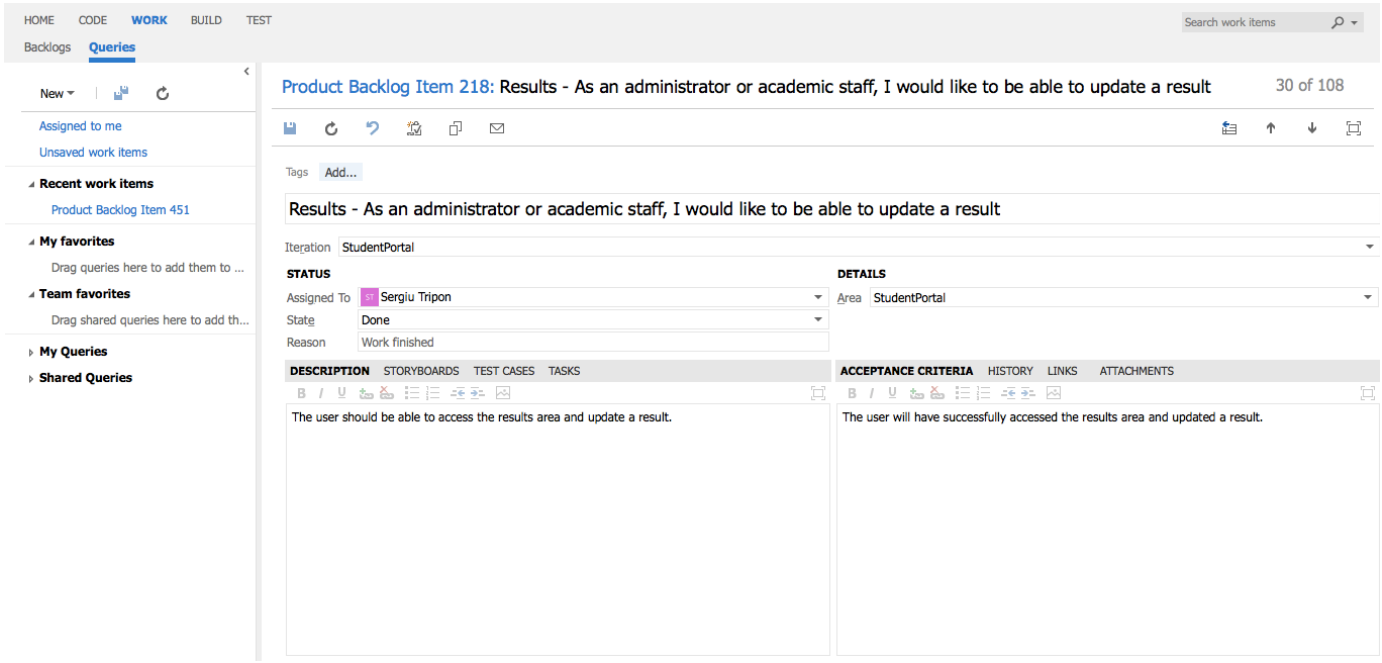
Events - detailed view of requirement



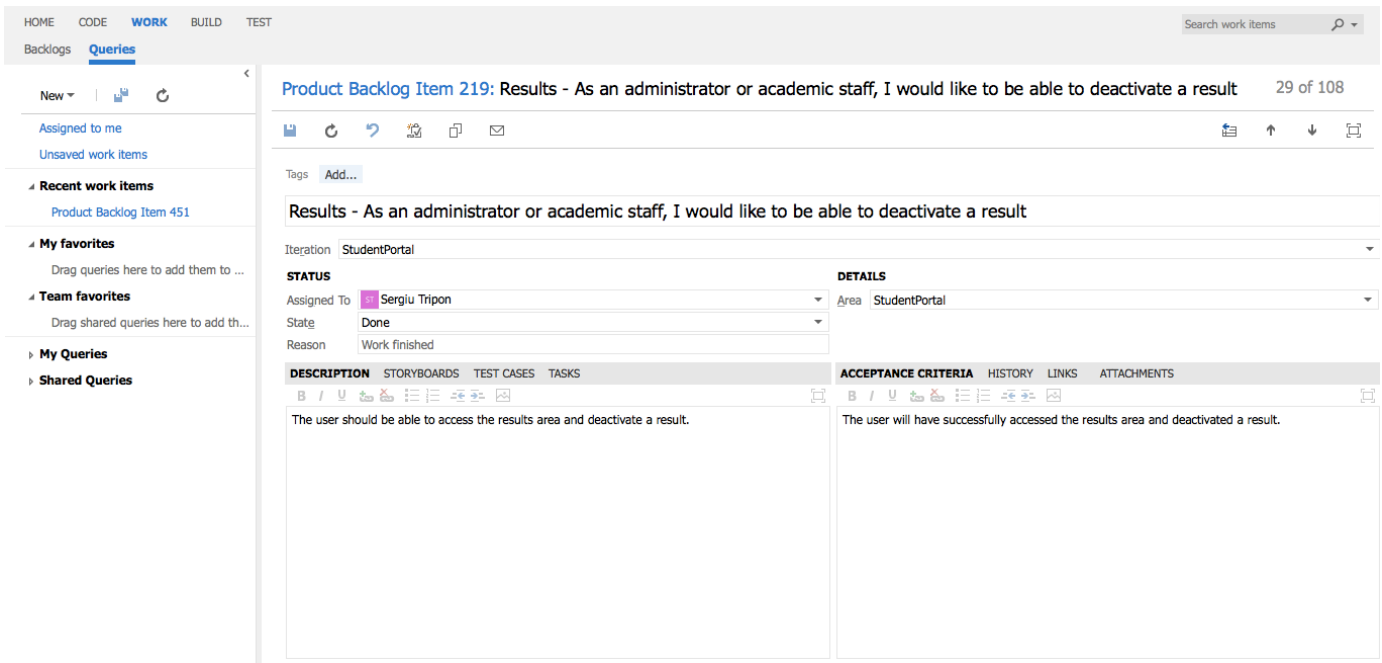
Results - detailed view of requirement



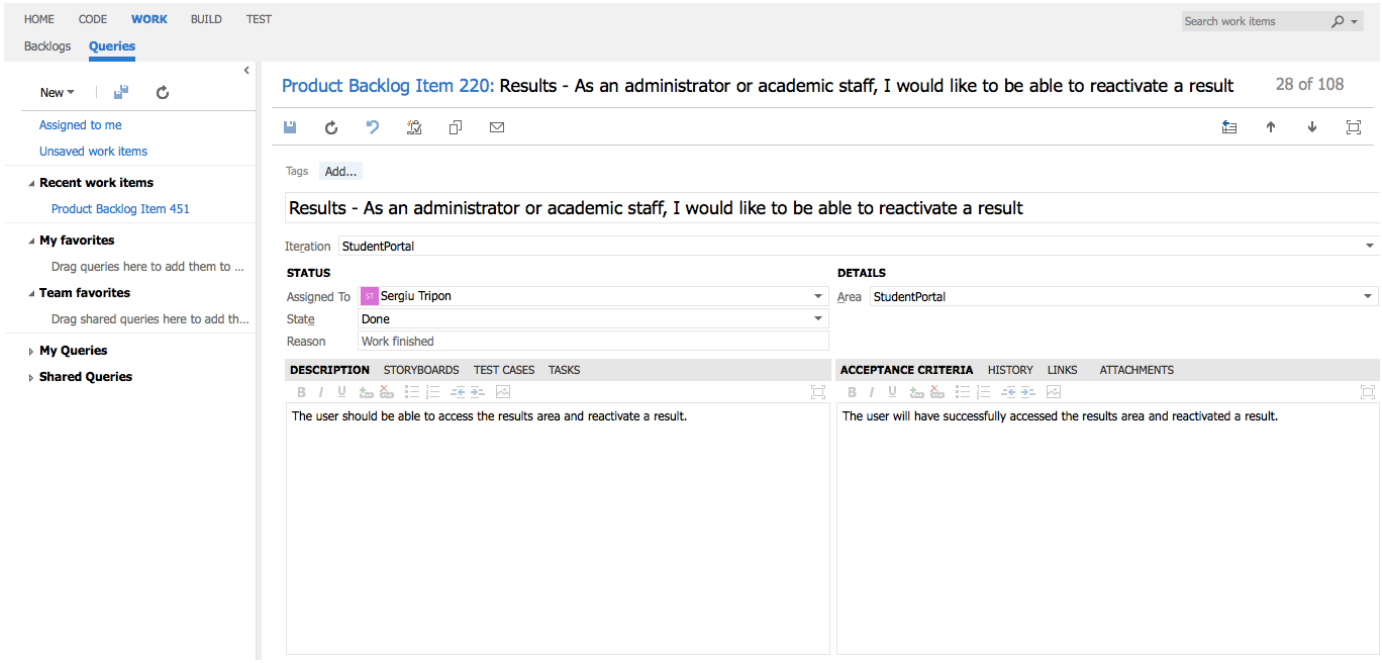
Results - detailed view of requirement



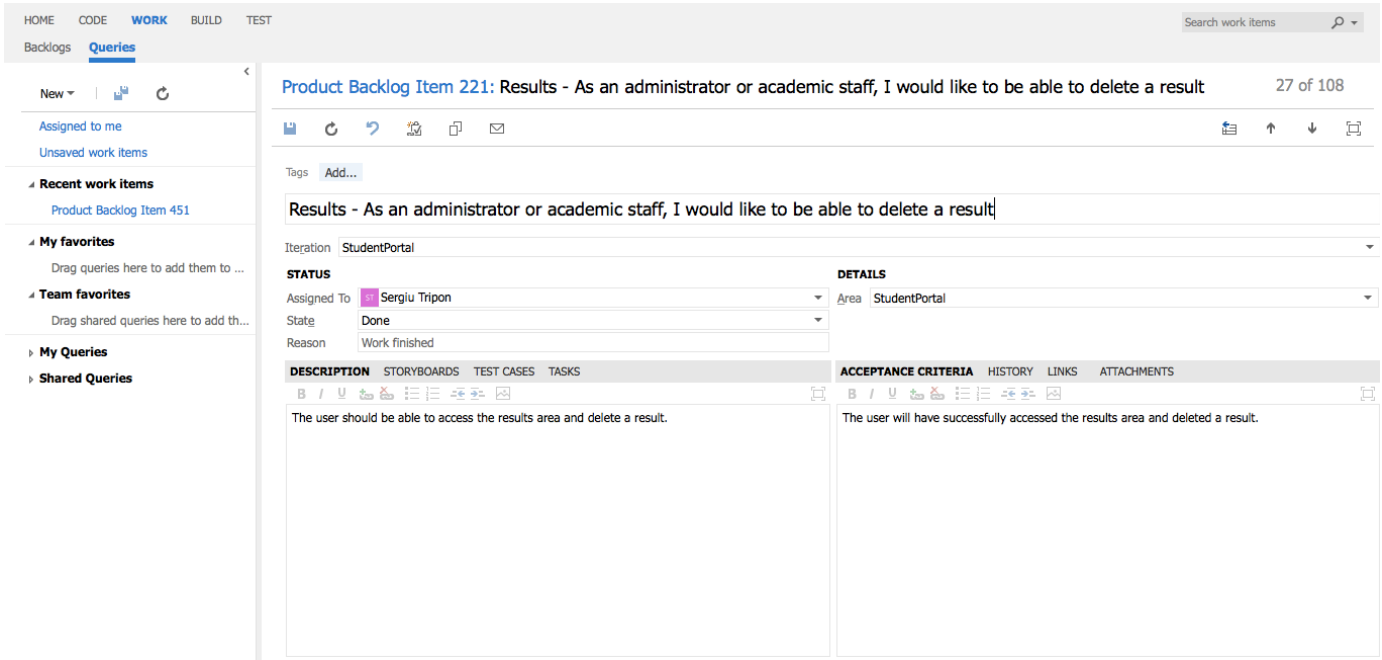
Results - detailed view of requirement



Results - detailed view of requirement



Results - detailed view of requirement



City University London

IN3007 Individual Project
Final Year Project Report

Academic Year: 2014-15

Appendix C
Analysis and Design document

by

Sergiu Tripon

Project supervisor: Dr. Vladimir Stankovic

Table of Contents

1. Use case specification

Sign In/Sign Out

Use case 1 (Sign In/Home page - Any account type - Sign In/Sign Out)

1. Brief description
This use case describes how the user will use Student Portal to sign in.

2. Actors
2.1 User (any type)
2.2 System

3. Preconditions
3.1 The user has internet connection.
3.2 The user has an e-mail address.
3.3 The user has an internet browser installed.
3.4 The user has an account.

4. Basic flow of events
4.1 The user will launch an internet browser
4.2 The user will type in the URL of Student Portal in the address bar
4.3 The system will display a “Sign In” form
4.4 The user will fill in the form
4.5 The user will click/tap the “Sign In” button
4.6 The system will authenticate the user
4.7 The system will redirect and display the “Home” page

5. Alternative flows
None

6. Key Scenarios
None

7. Post-conditions
7.1 Successful completion
The user has signed into the system.
7.2 Failure condition
None

Register

Use case 1 (Register - Student - Register)

1. Brief description
This use case describes how the user will use Student Portal to register.

2. Actors
2.1 User (account type: student)
2.2 System

3. Preconditions
3.1 The user has internet connection.
3.2 The user has an e-mail address.
3.3 The user has an internet browser installed.

4. Basic flow of events
4.1 The user will launch an internet browser
4.2 The user will type in the URL of Student Portal in the address bar
4.3 The system will display a “Register” form
4.4 The user will fill in the form
4.5 The user will click/tap the “Register” button
4.6 The system will register the user on the system.

5. Alternative flows
None

6. Key Scenarios
None

7. Post-conditions
7.1 Successful completion
The user is registered on the system.
7.2 Failure conditions
None

Forgotten your password/Password Reset

Use case 1 (Forgotten Password - Any account type - Reset password)

1. Brief description
This use case describes how the user will use Student Portal to reset their password which they have forgotten.

2. Actors
2.1 User (account type: any)
2.2 System

3. Preconditions
3.1 The user has internet connection.
3.2 The user has an e-mail address.
3.3 The user has an internet browser installed.

4. Basic flow of events
4.1 The user will launch an internet browser
4.2 The user will type in the URL of Student Portal in the address bar
4.3 The system will display a “Sign In” form
4.4 The user will click/tap on the link labelled “Forgotten your password?”
4.5 The system will load the “Forgotten your password” page
4.6 The system will display a “Forgotten your password” form
4.7 The user will fill in the form
4.8 The user will click/tap the “Continue” button.
4.9 The system will send an e-mail to the e-mail address provided containing a URL that directs the user to the “Password Reset” page order for the user to reset their password.

5. Alternative flows
None

6. Key Scenarios
None

7. Post-conditions
7.1 Successful completion
The user has passed the e-mail verification and has received an e-mail with password reset instructions.

7.2 Failure condition
None

Use case 2 (Password Reset - Any account type - Reset password)

1. Brief description
This use case describes how the user will use Student Portal to reset their password that they have forgotten.

2. Actors
2.1 User (any type)
2.2 System

3. Preconditions
3.1 The user has internet connection.
3.2 The user has e-mail address.
3.3 The user has an internet browser installed.

4. Basic flow of events
4.1 The user will check their e-mail account used to register on the Student Portal
4.2 The user will view the e-mail sent by Student Portal
4.3 The user will click on the link within the e-mail
4.4 The user's default internet browser will launch and load the "Password Reset" page
4.5 The system will load up a "Password Reset" form
4.6 The user will fill in the form
4.7 The user will click/tap the "Continue" button

5. Alternative flows
None

6. Key Scenarios
None

7. Post-conditions
7.1 Successful completion
The user has reset their password to a new one, and is able to sign into the system using the new password.
7.2 Failure condition
None

Timetable

Use case 1 (Timetable - Student - Check lecture and tutorial to attend)

1. Brief description
This use case describes how the user will use Student Portal to check lectures and tutorial they need to attend.

2. Actors
2.1 User (account type: student)
2.2 System

3. Preconditions
3.1 The user has internet connection.
3.2 The user has an e-mail address.
3.3 The user has an internet browser installed.
3.4 The user has an account.
3.5 The user has signed into the system.

4. Basic flow of events
4.1 The system loads the "Home" page
4.2 The user clicks/taps on "Timetable"
4.3 The system will load up a timetable that belongs to the currently signed in user
4.4 The user will click on the Lecture or Tutorial titles to view more detailed information about their upcoming lectures and tutorials that they have to attend. The user will be able to click on the "Moodle" button which will load up that specific module's Moodle page in a new tab within the web browser

5. Alternative flows
5.1 System fails to retrieve data
The system shall display an error message.
The use case resumes at step 4.2.

6. Key Scenarios
6.1 System fails to retrieve data

7. Post-conditions
7.1 Successful completion
The user has checked their timetable.

7.2 Failure condition
The system displays failure reasons accordingly.

Use case 2 (Timetable - Academic staff - Check lectures and tutorials to teach)

1. Brief description
This use case describes how the user will use Student Portal to check which lectures and tutorial they need to teach.

2. Actors
2.1 User (account type: academic staff)
2.2 System

3. Preconditions
3.1 The user has internet connection.
3.2 The user has an e-mail address.
3.3 The user has an internet browser installed.
3.4 The user has an account.
3.5 The user has signed into the system.

4. Basic flow of events
4.1 The system loads the "Home" page
4.2 The user clicks/tap on "Timetable"
4.3 The system will load up a timetable that belongs to the currently signed in user
4.4 The user will click on the Lecture or Tutorial titles to view more detailed information about their upcoming lectures and tutorials that they have to teach. The user will be able to click on the "Moodle" button which will load up that specific module's Moodle page in a new tab within the web browser

5. Alternative flows
5.1 System fails to retrieve data
The system shall display an error message.
The use case resumes at step 2.

6. Key Scenarios
6.1 System fails to retrieve data

7. Post-conditions
7.1 Successful completion
The user has checked their timetable.

7.2 Failure condition
The system displays failure reasons accordingly.

Use case 3 (Timetable - Administrator - Create a module/lecture/tutorial)

1. Brief description
This use case describes how the user will use Student Portal to create a module/lecture/tutorial.

2. Actors
2.1 User (account type: administrator)
2.2 System

3. Preconditions
3.1 The user has internet connection.
3.2 The user has an e-mail address.
3.3 The user has an internet browser installed.
3.4 The user has an account.
3.5 The user has signed into the system.

4. Basic flow of events
4.1 The system will load the “Home” page
4.2 The user will click/tap on “Timetable”
4.3 The user will click/tap on “Create a module/lecture/tutorial” button
4.4 The system will load up a “Create a module/lecture/tutorial” form
4.5 The user will fill in the form
4.6 The user will click on the “Create module/lecture/tutorial” button
4.7 The system will create the module/lecture/tutorial

5. Alternative flows
None

6. Key Scenarios
None

7. Post-conditions
7.1 Successful completion
The user has created a module/lecture/tutorial.

7.2 Failure condition
None

Use case 4 (Timetable - Administrator - Update a module/lecture/tutorial)

1. Brief description
This use case describes how the user will use Student Portal to update a module/lecture/tutorial.

2. Actors
2.1 User (account type: administrator)
2.2 System

3. Preconditions
3.1 The user has internet connection.
3.2 The user has an e-mail address.
3.3 The user has an internet browser installed.
3.4 The user has an account.
3.5 The user has signed into the system.

4. Basic flow of events
4.1 The system will load the "Home" page
4.2 The user will click/tap on "Timetable"
4.3 The system will load up already existing modules/lectures/tutorials
4.4 The user will click/tap the "Update module/lecture/tutorial" button next to a module/lecture/tutorial
4.5 The system will load up an "Update module/lecture/tutorial" form
4.6 The user will overwrite the form
4.7 The user will click/tap the "Update module/lecture/tutorial" button
4.8 The system will update the module/lecture/tutorial selected

5. Alternative flows
5.1 System fails to retrieve data
The system shall display an error message.
The use case resumes at step 5.

6. Key Scenarios
6.1 System fails to retrieve data

7. Post-conditions
7.1 Successful completion
The user has updated the module/lecture/tutorial selected.

7.2 Failure condition
The system displays failure reasons accordingly.

Use case 5 (Timetable - Administrator - Deactivate a module/lecture/tutorial)

1. Brief description
This use case describes how the user will use Student Portal to deactivate a module/lecture/tutorial.

2. Actors
2.1 User (account type: administrator)
2.2 System

3. Preconditions
3.1 The user has internet connection
3.2 The user has an e-mail address
3.3 The user has an internet browser installed
3.4 The user has an account
3.5 The user has signed into the system

4. Basic flow of events
4.1 The system will load the "Home" page
4.2 The user will click/tap on "Timetable"
4.3 The system will load up already existing modules/lectures/tutorials
4.4 The user will click/tap the "Deactivate module/lecture/tutorial" button next to a module/lecture/tutorial
4.5 The system will deactivate the module/lecture/tutorial selected

5. Alternative flows
5.1 System fails to retrieve data
The system shall display an error message
The use case resumes at step 5.

6. Key Scenarios
6.1 System fails to retrieve data

7. Post-conditions
7.1 Successful completion
The user has deactivated the module/lecture/tutorial selected.

7.2 Failure condition
The system displays failure reasons accordingly.

Use case 6 (Timetable - Administrator - Reactivate a module/lecture/tutorial)

1. Brief description
This use case describes how the user will use Student Portal to reactivate a module/lecture/tutorial.

2. Actors
2.1 User (account type: administrator)
2.2 System

3. Preconditions
3.1 The user has internet connection.
3.2 The user has an e-mail address.
3.3 The user has an internet browser installed.
3.4 The user has an account.
3.5 The user has signed into the system.

4. Basic flow of events
4.1 The system will load the "Home" page
4.2 The user will click/tap on "Timetable"
4.3 The system will load up already existing modules/lectures/tutorials
4.4 The user will click/tap the "Reactivate module/lecture/tutorial" button next to a module/lecture/tutorial
4.5 The system will reactivate the module/lecture/tutorial selected

5. Alternative flows
5.1 System fails to retrieve data
The system shall display an error message
The use case resumes at step 5.

6. Key Scenarios
6.1 System fails to retrieve data

7. Post-conditions
7.1 Successful completion
The user has reactivated the module/lecture/tutorial selected.

7.2 Failure condition
The system displays failure reasons accordingly.

Use case 7 (Timetable - Administrator - Delete a module/lecture/tutorial)

1. Brief description
This use case describes how the user will use Student Portal to delete a module/lecture/tutorial.

2. Actors
2.1 User (account type: administrator)
2.2 System

3. Preconditions
3.1 The user has internet connection.
3.2 The user has an e-mail address.
3.3 The user has an internet browser installed.
3.4 The user has an account.
3.5 The user has signed into the system.

4. Basic flow of events
4.1 The system will load the "Home" page
4.2 The user will click/tap on "Timetable"
4.3 The system will load up already existing modules/lectures/tutorials
4.4 The user will click/tap the "Delete module/lecture/tutorial" button next to a module/lecture/tutorial
4.5 The system will display a confirmation pop-up
4.6 The user will click/tap the "Delete" button
4.7 The system will delete the module/lecture/tutorial selected

5. Alternative flows
5.1 System fails to retrieve data
The system shall display an error message
The use case resumes at step 5.

6. Key Scenarios
6.1 System fails to retrieve data

7. Post-conditions
7.1 Successful completion
The user has deleted the module/lecture/tutorial selected.

7.2 Failure condition
The system displays failure reasons accordingly.

Exams

Use case 1 (Exams - Student - Check the exam timetable)

1. Brief description
This use case describes how the user will use Student Portal to check the exam timetable.

2. Actors
2.1 User (account type: student)
2.2 System

3. Preconditions
3.1 The user has internet connection.
3.2 The user has an e-mail address.
3.3 The user has an internet browser installed.
3.4 The user has an account.
3.5 The user has signed into the system.

4. Basic flow of events
4.1 The system will load the "Home" page
4.2 The user will click/tap on "Exams"
4.3 The user will click/tap on "Exam timetable"
4.4 The system will display a list of exams that the user is undertaking
4.5 The user will click on the Exam titles to view detailed information about their upcoming exams

5. Alternative flows
5.1 System fails to retrieve data
The system shall display an error message.
The use case resumes at step 2.

6. Key Scenarios
6.1 System fails to retrieve data

7. Post-conditions
7.1 Successful completion
The user has checked the exams timetable.
7.2 Failure condition
The system displays failure reasons accordingly.

Use case 2 (Exams - Academic staff - Check the exams related to modules taught)

1. Brief description
This use case describes how the user will use Student Portal to check the exams timetable in order to see exams related to the modules taught by the user.

2. Actors
2.1 User (account type: academic staff)
2.2 System

3. Preconditions
3.1 The user has internet connection.
3.2 The user has an e-mail address.
3.3 The user has an internet browser installed.
3.4 The user has an account.
3.5 The user has signed into the system.

4. Basic flow of events
4.1 The system will load the "Home" page
4.2 The user will click/tap on "Exams"
4.3 The user will click/tap on "Exam timetable"
4.4 The system will display a list of exams related to the modules taught by the user
4.5 The user will click on the Exam titles to view detailed information about the exams related to the modules taught by the user

5. Alternative flows
5.1 System fails to retrieve data
The system shall display an error message.
The use case resumes at step 2.

6. Key Scenarios
6.1 System fails to retrieve data

7. Post-conditions
7.1 Successful completion
The user has checked the exams related to the modules taught by the user.
7.2 Failure condition
The system displays failure reasons accordingly.

Use case 3 (Exams - Administrator - Create an exam)

1. Brief description
This use case describes how the user will use Student Portal to create an exam.

2. Actors
2.1 User (account type: administrator)
2.2 System

3. Preconditions
3.1 The user has internet connection.
3.2 The user has an e-mail address.
3.3 The user has an internet browser installed.
3.4 The user has an account.
3.5 The user has signed into the system.

4. Basic flow of events
4.1 The system will load the “Home” page
4.2 The user will click/tap on “Exams”
4.3 The user will click/tap on “Create an exam”
4.4 The system will load up a “Create an exam” form
4.5 The user will fill in the form
4.6 The user will click on the “Create exam” button
4.7 The system will create the exam

5. Alternative flows
None

6. Key Scenarios
None

7. Post-conditions
7.1 Successful completion
The user has created an exam.
7.2 Failure condition
None

Use case 4 (Exams - Administrator - Update an exam)

1. Brief description
This use case describes how the user will use Student Portal to update an exam.

2. Actors
2.1 User (account type: administrator)
2.2 System

3. Preconditions
3.1 The user has internet connection.
3.2 The user has an e-mail address.
3.3 The user has an internet browser installed.
3.4 The user has an account.
3.5 The user has signed into the system.

4. Basic flow of events
4.1 The system will load the “Home” page
4.2 The user will click/tap on “Exams”
4.3 The system will load up already existing exams
4.4 The user will click/tap the “Update” button next to an exam
4.5 The system will load up an “Update exam” form
4.6 The user will overwrite the form
4.7 The user will click/tap on the “Update exam” button
4.8 The system will update the exam selected

5. Alternative flows
5.1 System fails to retrieve data
The system shall display an error message.
The use case resumes at step 3.

6. Key Scenarios
6.1 System fails to retrieve data

7. Post-conditions
7.1 Successful completion
The user has updated the exam selected.
7.2 Failure condition
The system displays failure reasons accordingly.

Use case 5 (Exams - Administrator - Deactivate an exam)

1. Brief description
This use case describes how the user will use Student Portal to deactivate an exam.

2. Actors
2.1 User (account type: administrator)
2.2 System

3. Preconditions
3.1 The user has internet connection.
3.2 The user has an e-mail address.
3.3 The user has an internet browser installed.
3.4 The user has an account.
3.5 The user has signed into the system.

4. Basic flow of events
4.1 The system will load the "Home" page
4.2 The user will click/tap on "Exams"
4.3 The system will load up already existing exams
4.4 The user will click/tap the "Deactivate" button next to an exam
4.5 The system will deactivate the exam selected

5. Alternative flows
5.1 System fails to retrieve data
The system shall display an error message.
The use case resumes at step 5.

6. Key Scenarios
6.1 System fails to retrieve data

7. Post-conditions
7.1 Successful completion
The user has deactivated the exam selected.

7.2 Failure condition
The system displays failure reasons accordingly.

Use case 6 (Exams - Administrator - Reactivate an exam)

1. Brief description
This use case describes how the user will use Student Portal to reactivate an exam.

2. Actors
2.1 User (account type: administrator)
2.2 System

3. Preconditions
3.1 The user has internet connection.
3.2 The user has an e-mail address.
3.3 The user has an internet browser installed.
3.4 The user has an account.
3.5 The user has signed into the system.

4. Basic flow of events
4.1 The system will load the “Home” page
4.2 The user will click/tap on “Exams”
4.3 The system will load up already existing exams
4.4 The user will click/tap the “Reactivate” button next to an exam
4.5 The system will reactivate the exam selected

5. Alternative flows
5.1 System fails to retrieve data
The system shall display an error message.
The use case resumes at step 5.

6. Key Scenarios
6.1 System fails to retrieve data

7. Post-conditions
7.1 Successful completion
The user has reactivated the exam selected.

7.2 Failure condition
The system displays failure reasons accordingly.

Use case 7 (Exams - Administrator - Delete an exam)

1. Brief description
This use case describes how the user will use Student Portal to delete an exam.

2. Actors
2.1 User (account type: administrator)
2.2 System

3. Preconditions
3.1 The user has internet connection.
3.2 The user has an e-mail address.
3.3 The user has an internet browser installed.
3.4 The user has an account.
3.5 The user has signed into the system.

4. Basic flow of events
4.1 The system will load the "Home" page
4.2 The user will click/tap on "Exams"
4.3 The system will load up already existing exams
4.4 The user will click/tap the "Delete" button next to an exam
4.5 The system will display a confirmation pop-up
4.6 The user will click/tap the "Delete" button
4.7 The system will delete the exam selected

5. Alternative flows
5.1 System fails to retrieve data
The system shall display an error message.
The use case resumes at step 5.

6. Key Scenarios
6.1 System fails to retrieve data

7. Post-conditions
7.1 Successful completion
The user has deleted the exam selected.

7.2 Failure condition
The system displays failure reasons accordingly.

Results

Use case 1 (Results - Student - Check results)

1. Brief description
This use case describes how the user will use Student Portal to check their results.

2. Actors
2.1 User (account type: student)
2.2 System

3. Preconditions
3.1 The user has internet connection.
3.2 The user has an e-mail address.
3.3 The user has an internet browser installed.
3.4 The user has an account.
3.5 The user has signed into the system.

4. Basic flow of events
4.1 The system will load the “Home” page
4.2 The user will click/tap on “Results”
4.3 The system will display a list of results that the user has been awarded
4.4 The user will click on the Module titles to view detailed information about their results

5. Alternative flows
5.1 System fails to retrieve data
The system shall display an error message.
The use case resumes at step 2.

6. Key Scenarios
6.1 System fails to retrieve data

7. Post-conditions
7.1 Successful completion
The user has checked the results.
7.2 Failure condition
The system displays failure reasons accordingly.

Use case 2 (Results - Administrator/Academic staff - Create a result)

1. Brief description
This use case describes how the user will use Student Portal to create a result.

2. Actors
2.1 User (account type: administrator, academic staff)
2.2 System

3. Preconditions
3.1 The user has internet connection.
3.2 The user has an e-mail address.
3.3 The user has an internet browser installed.
3.4 The user has an account.
3.5 The user has signed into the system.

4. Basic flow of events
4.1 The system will load the "Home" page
4.2 The user will click/tap on "Results"
4.3 The user will click/tap on the "Select" button next to a user
4.4 The user will click/tap on the "Create" button next to a module
4.5 The user will click/tap on "Create result"
4.6 The system will load up a "Create result" form
4.7 The user will fill in the form
4.8 The user will click on the "Create result" button
4.9 The system will create the result

5. Alternative flows
None

6. Key Scenarios
None

7. Post-conditions
7.1 Successful completion
The user has created a result.
7.2 Failure condition
None

Use case 3 (Exams - Administrator/Academic staff - Update a result)

1. Brief description
This use case describes how the user will use Student Portal to update a result.

2. Actors
2.1 User (account type: administrator, academic staff)
2.2 System

3. Preconditions
3.1 The user has internet connection.
3.2 The user has an e-mail address.
3.3 The user has an internet browser installed.
3.4 The user has an account.
3.5 The user has signed into the system.

4. Basic flow of events
4.1 The system will load the "Home" page
4.2 The user will click/tap on "Results"
4.3 The user will click/tap on the "Select" button next to a user
4.4 The system will load up already existing results
4.5 The user will click/tap the "Update" button next to a result
4.6 The system will load up an "Update result" form
4.7 The user will overwrite the form
4.8 The user will click/tap on the "Update result" button
4.9 The system will update the result selected

5. Alternative flows
None

6. Key Scenarios
None

7. Post-conditions
7.1 Successful completion
The user has updated the result selected.
7.2 Failure condition
The system displays failure reasons accordingly.

Use case 4 (Results - Administrator/Academic staff - Deactivate a result)

1. Brief description
This use case describes how the user will use Student Portal to deactivate a result.

2. Actors
2.1 User (account type: administrator, academic staff)
2.2 System

3. Preconditions
3.1 The user has internet connection.
3.2 The user has an e-mail address.
3.3 The user has an internet browser installed.
3.4 The user has an account.
3.5 The user has signed into the system.

4. Basic flow of events
4.1 The system will load the "Home" page
4.2 The user will click/tap on "Results"
4.3 The user will click/tap on the "Select" button next to a user
4.4 The system will load up already existing results
4.5 The user will select a result
4.6 The user will click/tap the "Deactivate" button
4.7 The system will deactivate the result

5. Alternative flows
5.1 System fails to retrieve data
The system shall display an error message.
The use case resumes at step 5.

6. Key Scenarios
6.1 System fails to retrieve data

7. Post-conditions
7.1 Successful completion
The user has deactivated the result selected.

7.2 Failure condition
The system displays failure reasons accordingly.

Use case 5 (Exams - Administrator/Academic staff - Reactivate a result)

1. Brief description
This use case describes how the user will use Student Portal to reactivate a result.

2. Actors
2.1 User (account type: administrator, academic staff)
2.2 System

3. Preconditions
3.1 The user has internet connection.
3.2 The user has an e-mail address.
3.3 The user has an internet browser installed.
3.4 The user has an account.
3.5 The user has signed into the system.

4. Basic flow of events
4.1 The system will load the "Home" page
4.2 The user will click/tap on "Results"
4.3 The user will click/tap on the "Select" button next to a user
4.4 The system will load up already existing results
4.5 The user will select a result
4.6 The user will click/tap the "Reactivate" button
4.7 The system will reactivate the result

5. Alternative flows
5.1 System fails to retrieve data
The system shall display an error message.
The use case resumes at step 5.

6. Key Scenarios
6.1 System fails to retrieve data

7. Post-conditions
7.1 Successful completion
The user has reactivated the result selected.

7.2 Failure condition
The system displays failure reasons accordingly.

Use case 6 (Results - Administrator/Academic staff - Delete a result)

1. Brief description
This use case describes how the user will use Student Portal to delete a result.

2. Actors
2.1 User (account type: administrator, academic staff)
2.2 System

3. Preconditions
The user has internet connection.
The user has an e-mail address.
The user has an internet browser installed.
The user has an account.
The user has signed into the system.

4. Basic flow of events
4.1 The system will load the “Home” page
4.2 The user will click/tap on “Results”
4.3 The user will click/tap on the “Select” button next to a user
4.4 The system will load up already existing results
4.5 The user will click/tap the “Delete” button next to a result
4.6 The system will display a confirmation pop-up
4.7 The user will click/tap the “Delete” button
4.8 The system will delete the result

5. Alternative flows
5.1 System fails to retrieve data
The system shall display an error message.
The use case resumes at step 5.

6. Key Scenarios
6.1 System fails to retrieve data

7. Post-conditions
7.1 Successful completion
The user has deleted the result selected.

7.2 Failure condition
The system displays failure reasons accordingly.

Transport

Use case 1 (Transport - Any account type - Check transport status)

1. Brief description
This use case describes how the user will use Student Portal to check transport status. This includes Live and Weekend updates of Tube lines and stations and Cycle Hire updates.

2. Actors
2.1 User (account type: any)
2.2 System

3. Preconditions
3.1 The user has internet connection.
3.2 The user has an e-mail address.
3.3 The user has an internet browser installed.
3.4 The user has an account.
3.5 The user has signed into the system.

4. Basic flow of events
4.1 The system will load the "Overview" page
4.2 The user will click/tap on "Transport"
4.3 The system will display a list or "wall" of panels. The first four panels will be for the user's choice, as they will be labelled "Tube - Now", "Tube - This Weekend", "Tube Map" and "Cycle Hire – Availability Updates". The other panels represent each line within the London Tube network and its current status at the point of when the user will access it
4.4 The user will select an option.
4.5 The system will load the user's choice
4.6 The user will view the transport information

5. Alternative flows
5.1 System fails to retrieve data from TFL (Transport for London)
The system shall write an error message in the log file.
The use case resumes at step 2.

6. Key Scenarios
6.1 System fails to retrieve data from TFL (Transport for London)

7. Post-conditions
7.1 Successful completion
The user has checked the transport status successfully.

7.2 Failure condition
The system displays failure reasons accordingly.

Library

Use case 1 (Library - Student/Academic staff - Browse the book catalogue)

1. Brief description
This use case describes how the user will use Student Portal to browse the library book catalogue.

2. Actors
2.1 User (account type: student, academic staff)
2.2 System

3. Preconditions
3.1 The user has internet connection.
3.2 The user has an e-mail address.
3.3 The user has an internet browser installed.
3.4 The user has an account.
3.5 The user has signed into the system.

4. Basic flow of events
4.1 The system loads the "Overview" page
4.2 The user will click/tap on "Library"
4.3 The system will load up a list of books available to be reserved
4.4 The user will view the books available to be reserved.

5. Alternative flows
5.1 System fails to retrieve data
The system shall display an error message.
The use case resumes at step 4.2.

6. Key Scenarios
6.1 System fails to retrieve data

7. Post-conditions
7.1 Successful completion
The user has successfully browsed the book catalogue.
7.2 Failure condition
The system displays failure reasons accordingly.

Use case 2 (Library - Student/Academic staff - Reserve a book)

1. Brief description
This use case describes how the user will use Student Portal to reserve a book in order to collect it from the library.

2. Actors
2.1 User (account type: student, academic staff)
2.2 System

3. Preconditions
3.1 The user has internet connection.
3.2 The user has an e-mail address.
3.3 The user has an internet browser installed.
3.4 The user has an account.
3.5 The user has signed into the system.

4. Basic flow of events
4.1 The system loads the "Overview" page
4.2 The user clicks/taps on "Library"
4.3 The system will load up a list of books available to reserve
4.4 The user will click/tap the "Reserve" button next to a book
4.5 The system will display a "Reserve book" form
4.6 The user will view the data displayed in the form
4.7 The user will click/tap the "Reserve book" button
4.8 The system will reserve the book
4.9 The system will send an e-mail confirmation to the user.
4.10 The system will send an e-mail confirmation to the administrator.

5. Alternative flows
5.1 System fails to retrieve data
The system shall display an error message.
The use case resumes at step 4.2.

6. Key Scenarios
6.1 System fails to retrieve data

7. Post-conditions
7.1 Successful completion
The user has successfully reserved the book selected.
7.2 Failure condition
The system displays failure reasons accordingly.

Use case 3 (Library - Student/Academic staff - Renew a book)

1. Brief description
This use case describes how the user will use Student Portal to renew a book that they've already loaned out.

2. Actors
2.1 User (account type: student, academic staff)
2.2 System

3. Preconditions
3.1 The user has internet connection.
3.2 The user has an e-mail address.
3.3 The user has an internet browser installed.
3.4 The user has an account.
3.5 The user has signed into the system.

4. Basic flow of events
4.1 The system loads the "Overview" page
4.2 The user clicks/taps on "Library"
4.3 The system will load up a list of books that are loaned out by the user
4.4 The user will click/tap the "Renew" button next to a book
4.5 The system will display a "Renew book" form
4.6 The user will view the data displayed in the form
4.7 The user will click/tap the "Renew book" button
4.8 The user will view the data displayed in the form
4.9 The system will reserve the book
4.10 The system will send an e-mail confirmation to the user.
4.11 The system will send an e-mail confirmation to the administrator.

5. Alternative flows
5.1 System fails to retrieve data
The system shall display an error message.
The use case resumes at step 4.2.

6. Key Scenarios
6.1 System fails to retrieve data

7. Post-conditions
7.1 Successful completion
The user has successfully reserved the book selected.
7.2 Failure condition
The system displays failure reasons accordingly.

Use case 4 (Library - Student/Academic staff - Request a book)

1. Brief description
This use case describes how the user will use Student Portal to request a book that is already loaned out by another user.

2. Actors
2.1 User (account type: student, academic staff)
2.2 System

3. Preconditions
3.1 The user has internet connection.
3.2 The user has an e-mail address.
3.3 The user has an internet browser installed.
3.4 The user has an account.
3.5 The user has signed into the system.

4. Basic flow of events
4.1 The system loads the "Overview" page
4.2 The user clicks/taps on "Library"
4.3 The system will load up a list of books that are already loaned out
4.4 The user will click/tap the "Request" button next to a book
4.5 The system will display a "Request book" form
4.6 The user will view the data displayed in the form
4.7 The user will click/tap the "Request book" button
4.8 The system will request the book
4.9 The system will send an e-mail confirmation to the user who loaned the book initially.
4.10 The system will send an e-mail confirmation to the administrator.

5. Alternative flows
5.1 System fails to retrieve data
The system shall display an error message.
The use case resumes at step 4.2.

6. Key Scenarios
6.1 System fails to retrieve data

7. Post-conditions
7.1 Successful completion
The user has successfully requested the book selected.
7.2 Failure condition
The system displays failure reasons accordingly.

Use case 5 (Library - Administrator - Create a book)

1. Brief description
This use case describes how the user will use Student Portal to create a book.

2. Actors
2.1 User (account type: administrator)
2.2 System

3. Preconditions
3.1 The user has internet connection.
3.2 The user has an e-mail address.
3.3 The user has an internet browser installed.
3.4 The user has an account.
3.5 The user has signed into the system.

4. Basic flow of events
4.1 The system will load the “Home” page
4.2 The user will click/tap on “Library”
4.3 The user will click/tap on “Create a book”
4.4 The system will load up a “Create a book” form
4.5 The user will fill in the form
4.6 The user will click on the “Create book” button
4.7 The system will create the book

5. Alternative flows
None

6. Key Scenarios
None

7. Post-conditions
7.1 Successful completion
The user has successfully created a book.
7.2 Failure condition
None

Use case 6 (Library - Administrator - Update a book)

1. Brief description
This use case describes how the user will use Student Portal to update a book.

2. Actors
2.1 User (account type: administrator)
2.2 System

3. Preconditions
3.1 The user has internet connection.
3.2 The user has an e-mail address.
3.3 The user has an internet browser installed.
3.4 The user has an account.
3.5 The user has signed into the system.

4. Basic flow of events
4.1 The system will load the "Home" page
4.2 The user will click/tap on "Library"
4.3 The system will load up a list of already existing books
4.4 The user will click/tap the "Update" button next to a book
4.5 The system will load up an "Update book" form
4.6 The user will overwrite the form
4.7 The user will click on the "Update book" button
4.8 The system will update the book selected

5. Alternative flows
5.1 System fails to retrieve data
The system shall display an error message.
The use case resumes at step 4.5.

6. Key Scenarios
6.1 System fails to retrieve data

7. Post-conditions
7.1 Successful completion
The user has successfully updated the book selected
7.2 Failure condition
The system displays failure reasons accordingly.

Use case 7 (Library - Administrator - Deactivate a book)

1. Brief description
This use case describes how the user will use Student Portal to deactivate a book.

2. Actors
2.1 User (account type: administrator)
2.2 System

3. Preconditions
3.1 The user has internet connection.
3.2 The user has an e-mail address.
3.3 The user has an internet browser installed.
3.4 The user has an account.
3.5 The user has signed into the system.

4. Basic flow of events
4.1 The system will load the "Home" page
4.2 The user will click/tap on "Library"
4.3 The system will load a list of already existing books
4.5 The user will click/tap the "Deactivate" button next to a book
4.6 The system will deactivate the book selected

5. Alternative flows
5.1 System fails to retrieve data
The system shall display an error message.
The use case resumes at step 4.2.

6. Key Scenarios
6.1 System fails to retrieve data

7. Post-conditions
7.1 Successful completion
The user has deactivated the book selected.
7.2 Failure condition
The system displays failure reasons accordingly.

Use case 8 (Library - Administrator - Reactivate a book)

1. Brief description
This use case describes how the user will use Student Portal to reactivate a book.

2. Actors
2.1 User (account type: administrator)
2.2 System

3. Preconditions
3.1 The user has internet connection.
3.2 The user has an e-mail address.
3.3 The user has an internet browser installed.
3.4 The user has an account.
3.5 The user has signed into the system.

4. Basic flow of events
4.1 The system will load the "Home" page
4.2 The user will click/tap on "Library"
4.3 The system will load a list of already existing books
4.5 The user will click/tap the "Reactivate" button next to a book
4.6 The system will deactivate the book selected

5. Alternative flows
5.1 System fails to retrieve data
The system shall display an error message.
The use case resumes at step 4.2.

6. Key Scenarios
6.1 System fails to retrieve data

7. Post-conditions
7.1 Successful completion
The user has reactivated the book selected.
7.2 Failure condition
The system displays failure reasons accordingly.

Use case 9 (Library - Administrator - Delete a book)

1. Brief description
This use case describes how the user will use Student Portal to delete a book.

2. Actors
2.1 User (account type: administrator)
2.2 System

3. Preconditions
3.1 The user has internet connection.
3.2 The user has an e-mail address.
3.3 The user has an internet browser installed.
3.4 The user has an account.
3.5 The user has signed into the system.

4. Basic flow of events
4.1 The system will load the “Home” page
4.2 The user will click/tap on “Library”
4.3 The system will load a list of already existing books
4.4 The user will click/tap the “Delete” button next to a book
4.5 The system will display a confirmation pop-up
4.6 The user will click/tap the “Delete” button
4.7 The system will delete the book

5. Alternative flows
5.1 System fails to retrieve data
The system shall display an error message.
The use case resumes at step 4.2.

6. Key Scenarios
6.1 System fails to retrieve data

7. Post-conditions
7.1 Successful completion
The user has deleted the book selected.
7.2 Failure condition
The system displays failure reasons accordingly.

Use case 10 (Library - Administrator - Mark book as collected)

1. Brief description
This use case describes how the user will use Student Portal to mark a book as collected.

2. Actors
2.1 User (account type: administrator)
2.2 System

3. Preconditions
3.1 The user has internet connection.
3.2 The user has an e-mail address.
3.3 The user has an internet browser installed.
3.4 The user has an account.
3.5 The user has signed into the system.

4. Basic flow of events
4.1 The system will load the "Home" page
4.2 The user will click/tap on "Library"
4.3 The system will load a list of books that are reserved
4.4 The user will click/tap the "Mark collected" button next to a book
4.5 The system will display a confirmation pop-up
4.6 The user will click/tap the "Mark collected" button
4.7 The system will mark the book selected as collected

5. Alternative flows
5.1 System fails to retrieve data
The system shall display an error message.
The use case resumes at step 4.2.

6. Key Scenarios
6.1 System fails to retrieve data

7. Post-conditions
7.1 Successful completion
The user has marked the book selected as collected.
7.2 Failure condition
The system displays failure reasons accordingly.

Use case 11 (Library - Administrator - Mark book as returned)

1. Brief description
This use case describes how the user will use Student Portal to mark a book as returned.

2. Actors
2.1 User (account type: administrator)
2.2 System

3. Preconditions
3.1 The user has internet connection.
3.2 The user has an e-mail address.
3.3 The user has an internet browser installed.
3.4 The user has an account.
3.5 The user has signed into the system.

4. Basic flow of events
4.1 The system will load the "Home" page
4.2 The user will click/tap on "Library"
4.3 The system will load a list of books that are loaned out
4.4 The user will click/tap the "Mark return" button next to a book
4.5 The system will display a confirmation pop-up
4.6 The user will click/tap the "Mark returned" button
4.7 The system will mark the book selected as returned

5. Alternative flows
5.1 System fails to retrieve data
The system shall display an error message.
The use case resumes at step 4.2.

6. Key Scenarios
6.1 System fails to retrieve data

7. Post-conditions
7.1 Successful completion
The user has marked the book selected as returned.
7.2 Failure condition
The system displays failure reasons accordingly.

Calendar

Use case 1 (Calendar - Any account type - Check tasks)

1. Brief description
This use case describes how the user will use Student Portal to check their due, completed or archived tasks.

2. Actors
2.1 User (account type: any)
2.2 System

3. Preconditions
3.1 The user has internet connection.
3.2 The user has an e-mail address.
3.3 The user has an internet browser installed.
3.4 The user has an account.
3.5 The user has signed into the system.

4. Basic flow of events
4.1 The system will load the “Home” page
4.2 The user will click/tap on “Calendar”
4.3 The system will load up lists displaying due, completed and archived tasks
4.4 The user will click on the Task titles to view detailed information about the tasks

5. Alternative flows
5.1 System fails to retrieve data
The system shall display an error message.
The use case resumes at step 4.2.

6. Key Scenarios
6.1 System fails to retrieve data

7. Post-conditions
7.1 Successful completion
The user has check their due, completed or archived tasks.
7.2 Failure condition
The system displays failure reasons accordingly.

Use case 2 (Calendar - Any account type - Check calendar)

1. Brief description
This use case describes how the user will use Student Portal to check their calendar.

2. Actors
2.1 User (account type: any)
2.2 System

3. Preconditions
3.1 The user has internet connection.
3.2 The user has an e-mail address.
3.3 The user has an internet browser installed.
3.4 The user has an account.
3.5 The user has signed into the system.

4. Basic flow of events
4.1 The system will load the "Home" page
4.2 The user will click/tap on "Calendar"
4.3 The system will load up a calendar displaying already existing tasks, booked events or loaned books
4.4 The user will view the information displayed on the calendar

5. Alternative flows
5.1 System fails to retrieve data
The system shall display an error message.
The use case resumes at step 4.2.

6. Key Scenarios
6.1 System fails to retrieve data

7. Post-conditions
7.1 Successful completion
The user has checked the information displayed on the calendar.
7.2 Failure condition
The system displays failure reasons accordingly.

Use case 3 (Calendar - Any account type - Create a task)

1. Brief description
This use case describes how the user will use Student Portal to create a task.

2. Actors
2.1 User (account type: any)
2.2 System

3. Preconditions
3.1 The user has internet connection.
3.2 The user has an e-mail address.
3.3 The user has an internet browser installed.
3.4 The user has an account.
3.5 The user has signed into the system.

4. Basic flow of events
4.1. The system will load the "Home" page
4.2 The user will click/tap on "Calendar"
4.3 The user will click/tap the "Create task" button
4.4 The system will load up a "Create a task" form
4.5 The user will fill in the form
4.6 The user will click/tap the "Create task" button
4.7 The system will create the task

5. Alternative flows
None

6. Key Scenarios
None

7. Post-conditions
7.1 Successful completion
The user has successfully created a task.
7.2 Failure condition
None

Use case 4 (Calendar - Any account type - Update a task)

1. Brief description
This use case describes how the user will use Student Portal to update a task.

2. Actors
2.1 User (account type: any)
2.2 System

3. Preconditions
3.1 The user has internet connection.
3.2 The user has an e-mail address.
3.3 The user has an internet browser installed.
3.4 The user has an account.
3.5 The user has signed into the system.

4. Basic flow of events
4.1 The system will load the “Overview” page
4.2 The user will click/tap on “Calendar”
4.3 The system will load up a list of already existing tasks
4.4 The user will click/tap the “Update task” button next to a task
4.5 The system will load up the “Update task” form
4.6 The user will overwrite the form
4.7 The user will click/tap the “Update task” button
4.8 The system will update the task selected.

5. Alternative flows
None

6. Key Scenarios
None

7. Post-conditions
7.1 Successful completion
The user has updated the task selected.
7.2 Failure condition
None

Use case 5 (Calendar - Any account type - Complete a task)

1. Brief description
This use case describes how the user will use Student Portal to complete a task.

2. Actors
2.1 User (account type: any)
2.2 System

3. Preconditions
3.1 The user has internet connection.
3.2 The user has an e-mail address.
3.3 The user has an internet browser installed.
3.4 The user has an account.
3.5 The user has signed into the system.

4. Basic flow of events
4.1 The system will load the "Home" page
4.2 The user will click/tap on "Calendar"
4.3 The system will load up a list of already existing tasks
4.4 The user will click/tap the "Complete" button next to a task
4.5 The system will complete the task selected.

5. Alternative flows
5.1 System fails to retrieve data
The system shall display an error message.
The use case resumes at step 4.2.

6. Key Scenarios
6.1 System fails to retrieve data

7. Post-conditions
7.1 Successful completion
The user has successfully completed the task selected.
7.2 Failure condition
The system displays failure reasons accordingly.

Use case 6 (Calendar - Any account type - Archive a task)

1. Brief description
This use case describes how the user will use Student Portal to archive a task.

2. Actors
2.1 User (account type: any)
2.2 System

3. Preconditions
3.1 The user has internet connection.
3.2 The user has an e-mail address.
3.3 The user has an internet browser installed.
3.4 The user has an account.
3.5 The user has signed into the system.

4. Basic flow of events
4.1 The system will load the "Home" page
4.2 The user will click/tap on "Calendar"
4.3 The system will load up a list of already existing tasks
4.4 The user will click/tap the "Archive" button next to a task
4.5 The system archive the task selected.

5. Alternative flows
5.1 System fails to retrieve data
The system shall display an error message.
The use case resumes at step 2.

6. Key Scenarios
6.1 System fails to retrieve data

7. Post-conditions
7.1 Successful completion
The user has archived the task selected.
7.2 Failure condition
The system displays failure reasons accordingly.

Use case 7 (Calendar - Any account type - Restore a task)

1. Brief description
This use case describes how the user will use Student Portal to restore a task.

2. Actors
2.1 User (account type: any)
2.2 System

3. Preconditions
3.1 The user has internet connection.
3.2 The user has an e-mail address.
3.3 The user has an internet browser installed.
3.4 The user has an account.
3.5 The user has signed into the system.

4. Basic flow of events
4.1 The system will load the “Home” page
4.2 The user will click/tap on “Calendar”
4.3 The system will load up a list of already existing tasks
4.4 The user will click/tap the “Restore” button next to a task
4.5 The system will restore the task selected.

5. Alternative flows
5.1 System fails to retrieve data
The system shall display an error message.
The use case resumes at step 2.

6. Key Scenarios
6.1 System fails to retrieve data

7. Post-conditions
7.1 Successful completion
The user has restored the task selected.
7.2 Failure condition
The system displays failure reasons accordingly.

Use case 8 (Calendar - Any account type - Delete a task)

1. Brief description
This use case describes how the user will use Student Portal to delete a task.

2. Actors
2.1 User (account type: any)
2.2 System

3. Preconditions
3.1 The user has internet connection.
3.2 The user has an e-mail address.
3.3 The user has an internet browser installed.
3.4 The user has an account.
3.5 The user has signed into the system.

4. Basic flow of events
4.1 The system will load the “Home” page
4.2 The user will click/tap on “Calendar”
4.3 The system will load up a list of already existing tasks
4.4 The user will click/tap the “Delete” button next to a task
4.5 The system will delete the task selected.

5. Alternative flows
5.1 System fails to retrieve data
The system shall display an error message.
The use case resumes at step 4.2.

6. Key Scenarios
6.1 System fails to retrieve data

7. Post-conditions
7.1 Successful completion
The user has deleted the task selected.
7.2 Failure condition
The system displays failure reasons accordingly.

University Map

Use case 1 (University Map - Student/Academic staff - Check the location overview)

1. Brief description
This use case describes how the user will use Student Portal to view an overview of the location surrounding the university campus

2. Actors
2.1 User (account type: student, academic staff)
2.2 System

3. Preconditions
3.1 The user has internet connection.
3.2 The user has an e-mail address.
3.3 The user has an internet browser installed.
3.4 The user has an account.
3.5 The user has signed into the system.

4. Basic flow of events
4.1 The system will load the “Home” page
4.2 The user will click/tap on “University Map”
4.3 The user will click/tap on “View all locations”
4.4 The system will load up a map display locations surrounding the university campus
4.5 The user will click on the map markers to view information about the locations

5. Alternative flows
5.1 Map load failure
The system shall display the message an error message.
The use case resumes at step 2.

6. Key Scenarios
None

7. Post-conditions
7.1 Successful completion
The user has checked the location overview.
7.2 Failure condition
None

Use case 2 (University Map - Student/Academic staff - Filter locations)

1. Brief description
This use case describes how the user will use Student Portal to filter the location displayed on the location overview by category.

2. Actors
2.1 User (account type: student, academic staff)
2.2 System

3. Preconditions
3.1 The user has internet connection.
3.2 The user has an e-mail address.
3.3 The user has an internet browser installed.
3.4 The user has an account.
3.5 The user has signed into the system.

4. Basic flow of events
4.1 The system will load the "Home" page
4.2 The user will click/tap on "University Map"
4.3 The user will click/tap on "View all locations"
4.4 The system will load up a map display locations surrounding the university campus
4.5 The user will tick and untick the category in order to filter the location displayed on the map by category

5. Alternative flows
5.1 Map load failure
The system shall display the message an error message.
The use case resumes at step 2.

6. Key Scenarios
None

7. Post-conditions
7.1 Successful completion
The user has filtered the locations on the location overview by category.
7.2 Failure condition
None

Use case 3 (University Map - Student/Academic staff - Search for a location)

1. Brief description
This use case describes how the user will use Student Portal to search for a location.

2. Actors
2.1 User (account type: student, academic staff)
2.2 System

3. Preconditions
3.1 The user has internet connection.
3.2 The user has an e-mail address.
3.3 The user has an internet browser installed.
3.4 The user has an account.
3.5 The user has signed into the system.

4. Basic flow of events
4.1 The system will load the “Home” page
4.2 The user will click/tap on “University Map”
4.3 The user will click/tap on “Search for a location”
4.4 The system will load up the “Search for a location” form
4.5 The user will fill in the form
4.6 The user will click/tap the “Search” button
4.7 The system will display the locations found on the map

5. Alternative flows
5.1 Map load failure
The system shall display the message an error message.
The use case resumes at step 2.

6. Key Scenarios
None

7. Post-conditions
7.1 Successful completion
The user has search for a location.
7.2 Failure condition
None

Use case 4 (University Map - Administrator - Create a location)

1. Brief description
This use case describes how the user will use Student Portal to create a location.

2. Actors
2.1 User (account type: administrator)
2.2 System

3. Preconditions
3.1 The user has internet connection.
3.2 The user has an e-mail address.
3.3 The user has an internet browser installed.
3.4 The user has an account.
3.5 The user has signed into the system.

4. Basic flow of events
4.1. The system will load the "Home" page
4.2 The user will click/tap on "University Map"
4.3 The user will click/tap the "Create location" button
4.4 The system will load up a "Create a location" form
4.5 The user will fill in the form
4.6 The user will click/tap the "Create location" button
4.7 The system will create the location

5. Alternative flows
None

6. Key Scenarios
None

7. Post-conditions
7.1 Successful completion
The user has created a location.
7.2 Failure condition
None

Use case 5 (University Map - Administrator - Update a location)

1. Brief description
This use case describes how the user will use Student Portal to update a location.

2. Actors
2.1 User (account type: administrator)
2.2 System

3. Preconditions
3.1 The user has internet connection.
3.2 The user has an e-mail address.
3.3 The user has an internet browser installed.
3.4 The user has an account.
3.5 The user has signed into the system.

4. Basic flow of events
4.1 The system will load the “Overview” page
4.2 The user will click/tap on “University Map”
4.3 The system will load up a list of already existing locations
4.4 The user will click/tap the “Update task” button next to a location
4.5 The system will load up the “Update location” form
4.6 The user will overwrite the form
4.7 The user will click/tap the “Update location” button
4.8 The system will update the location selected.

5. Alternative flows
None

6. Key Scenarios
None

7. Post-conditions
7.1 Successful completion
The user has updated the location selected.
7.2 Failure condition
None

Use case 6 (University Map - Administrator - Deactivate a location)

1. Brief description
This use case describes how the user will use Student Portal to deactivate a location.

2. Actors
2.1 User (account type: administrator)
2.2 System

3. Preconditions
3.1 The user has internet connection.
3.2 The user has an e-mail address.
3.3 The user has an internet browser installed.
3.4 The user has an account.
3.5 The user has signed into the system.

4. Basic flow of events
4.1 The system will load the "Home" page
4.2 The user will click/tap on "University Map"
4.3 The system will load up a list of already existing locations
4.4 The user will click/tap the "Deactivate" button next to a location
4.5 The system will deactivate the location selected.

5. Alternative flows
5.1 System fails to retrieve data
The system shall display an error message.
The use case resumes at step 4.2.

6. Key Scenarios
6.1 System fails to retrieve data

7. Post-conditions
7.1 Successful completion
The user has successfully deactivated the location selected.
7.2 Failure condition
The system displays failure reasons accordingly.

Use case 7 (University Map - Administrator - Reactivate a location)

1. Brief description
This use case describes how the user will use Student Portal to reactivate a location.

2. Actors
2.1 User (account type: administrator)
2.2 System

3. Preconditions
3.1 The user has internet connection.
3.2 The user has an e-mail address.
3.3 The user has an internet browser installed.
3.4 The user has an account.
3.5 The user has signed into the system.

4. Basic flow of events
4.1 The system will load the “Home” page
4.2 The user will click/tap on “University Map”
4.3 The system will load up a list of already existing locations
4.4 The user will click/tap the “Reactivate” button next to a location
4.5 The system archive the location selected.

5. Alternative flows
5.1 System fails to retrieve data
The system shall display an error message.
The use case resumes at step 2.

6. Key Scenarios
6.1 System fails to retrieve data

7. Post-conditions
7.1 Successful completion
The user has archived the location selected.
7.2 Failure condition
The system displays failure reasons accordingly.

Use case 8 (University Map - Administrator - Delete a location)

1. Brief description
This use case describes how the user will use Student Portal to delete a location.

2. Actors
2.1 User (account type: any)
2.2 System

3. Preconditions
3.1 The user has internet connection.
3.2 The user has an e-mail address.
3.3 The user has an internet browser installed.
3.4 The user has an account.
3.5 The user has signed into the system.

4. Basic flow of events
4.1 The system will load the "Home" page
4.2 The user will click/tap on "University Map"
4.3 The system will load up a list of already existing locations
4.4 The user will click/tap the "Delete" button next to a location
4.5 The system will delete the location selected.

5. Alternative flows
5.1 System fails to retrieve data
The system shall display an error message.
The use case resumes at step 4.2.

6. Key Scenarios
6.1 System fails to retrieve data

7. Post-conditions
7.1 Successful completion
The user has deleted the location selected.
7.2 Failure condition
The system displays failure reasons accordingly.

Events

Use case 1 (Events - Student/Academic staff - Check events calendar)

1. Brief description
This use case describes how the user will use Student Portal to check the events calendar.

2. Actors
2.1 User (account type: student, academic staff)
2.2 System

3. Preconditions
3.1 The user has internet connection.
3.2 The user has an e-mail address.
3.3 The user has an internet browser installed.
3.4 The user has an account.
3.5 The user has signed into the system.

4. Basic flow of events
4.1 The system will load the "Overview" page
4.2 The user will click/tap on "Events"
4.3 The system will load up a calendar displaying already existing events
4.4 The user will view the upcoming events displayed on the calendar

5. Alternative flows
5.1 System fails to retrieve data
The system shall display an error message.
The use case resumes at step 2.

6. Key Scenarios
6.1 System fails to retrieve data

7. Post-conditions
7.1 Successful completion
The user has checked the events calendar.
7.2 Failure condition
The system displays failure reasons accordingly.

Use case 2 (Events - Student/Academic staff - Book and Pay for an event)

1. Brief description
This use case describes how the user will use Student Portal to book and pay for an event.

2. Actors
2.1 User (account type: student, lecturer)
2.2 System

3. Preconditions
3.1 The user has internet connection.
3.2 The user has an e-mail address.
3.3 The user has an internet browser installed.
3.4 The user has an account.
3.5 The user has signed into the system.

4. Basic flow of events
4.1 The system will load the "Overview" page
4.2 The user will click/tap on "Events"
4.3 The system will load up already existing events
4.4 The user will click/tap the "Book" button next to an event
4.5 The system will display a "Book an event" form
4.6 The user will fill in the form
4.7 The user will clicks/tap the "Book" button
4.8 The system will redirect to PayPal
4.9 The user will complete the payment on PayPal
4.10 The system will send a confirmation e-mail to the user

5. Alternative flows
None

6. Key Scenarios
None

7. Post-conditions
7.1 Successful completion
The user has booked and paid for the event selected.
7.2 Failure condition
None

Use case 3 (Events - Administrator - Create an event)

1. Brief description
This use case describes how the user will use Student Portal to create an event.

2. Actors
2.1 User (account type: administrator)
2.2 System

3. Preconditions
3.1 The user has internet connection.
3.2 The user has an e-mail address.
3.3 The user has an internet browser installed.
3.4 The user has an account.
3.5 The user has signed into the system.

4. Basic flow of events
4.1. The system will load the "Home" page
4.2 The user will click/tap on "Events"
4.3 The user will click/tap the "Create event" button
4.4 The system will load up a "Create event" form
4.5 The user will fill in the form
4.6 The user will click/tap the "Create event" button
4.7 The system will create the event

5. Alternative flows
None

6. Key Scenarios
None

7. Post-conditions
7.1 Successful completion
The user has created an event.
7.2 Failure condition
None

Use case 4 (Events - Administrator - Update event)

1. Brief description
This use case describes how the user will use Student Portal to update an event.

2. Actors
2.1 User (account type: administrator)
2.2 System

3. Preconditions
3.1 The user has internet connection.
3.2 The user has an e-mail address.
3.3 The user has an internet browser installed.
3.4 The user has an account.
3.5 The user has signed into the system.

4. Basic flow of events
4.1 The system will load the “Overview” page
4.2 The user will click/tap on “Events”
4.3 The system will load up a list of already existing locations
4.4 The user will click/tap the “Update event” button next to a location
4.5 The system will load up the “Update event” form
4.6 The user will overwrite the form
4.7 The user will click/tap the “Update event” button
4.8 The system will update the event selected.

5. Alternative flows
None

6. Key Scenarios
None

7. Post-conditions
7.1 Successful completion
The user has updated the event selected.
7.2 Failure condition
None

Use case 5 (Events - Administrator - Deactivate a event)

1. Brief description
This use case describes how the user will use Student Portal to deactivate an event.

2. Actors
2.1 User (account type: administrator)
2.2 System

3. Preconditions
3.1 The user has internet connection.
3.2 The user has an e-mail address.
3.3 The user has an internet browser installed.
3.4 The user has an account.
3.5 The user has signed into the system.

4. Basic flow of events
4.1 The system will load the "Home" page
4.2 The user will click/tap on "Events"
4.3 The system will load up a list of already existing events
4.4 The user will click/tap the "Deactivate" button next to an event
4.5 The system will deactivate the event selected.

5. Alternative flows
5.1 System fails to retrieve data
The system shall display an error message.
The use case resumes at step 4.2.

6. Key Scenarios
6.1 System fails to retrieve data

7. Post-conditions
7.1 Successful completion
The user has successfully deactivated the event selected.
7.2 Failure condition
The system displays failure reasons accordingly.

Use case 6 (Events - Administrator - Reactivate event)

1. Brief description
This use case describes how the user will use Student Portal to reactivate a event.

2. Actors
2.1 User (account type: administrator)
2.2 System

3. Preconditions
3.1 The user has internet connection.
3.2 The user has an e-mail address.
3.3 The user has an internet browser installed.
3.4 The user has an account.
3.5 The user has signed into the system.

4. Basic flow of events
4.1 The system will load the "Home" page
4.2 The user will click/tap on "Events"
4.3 The system will load up a list of already existing events
4.4 The user will click/tap the "Reactivate" button next to an event
4.5 The system archive the event selected.

5. Alternative flows
5.1 System fails to retrieve data
The system shall display an error message.
The use case resumes at step 2.

6. Key Scenarios
6.1 System fails to retrieve data

7. Post-conditions
7.1 Successful completion
The user has archived the event selected.
7.2 Failure condition
The system displays failure reasons accordingly.

Use case 7 (Events - Administrator - Delete event)

1. Brief description
This use case describes how the user will use Student Portal to delete an event.

2. Actors
2.1 User (account type: any)
2.2 System

3. Preconditions
3.1 The user has internet connection.
3.2 The user has an e-mail address.
3.3 The user has an internet browser installed.
3.4 The user has an account.
3.5 The user has signed into the system.

4. Basic flow of events
4.1 The system will load the "Home" page
4.2 The user will click/tap on "Events"
4.3 The system will load up a list of already existing events
4.4 The user will click/tap the "Delete" button next to an event
4.5 The system will delete the event selected.

5. Alternative flows
5.1 System fails to retrieve data
The system shall display an error message.
The use case resumes at step 4.2.

6. Key Scenarios
6.1 System fails to retrieve data

7. Post-conditions
7.1 Successful completion
The user has deleted the event selected.
7.2 Failure condition
The system displays failure reasons accordingly.

Feedback

Use case 1 (Feedback - Student - Submit feedback for a module)

1. Brief description
This use case describes how the user will use Student Portal to submit feedback for a module.

2. Actors
2.1 User (account type: student)
2.2 System

3. Preconditions
3.1 The user has internet connection.
3.2 The user has an e-mail address.
3.3 The user has an internet browser installed.
3.4 The user has an account.
3.5 The user has signed into the system.

4. Basic flow of events
1. The system will load the “Overview” page
2. The user will click/tap on “Feedback”
3. The system will load up a list of modules
4. The user will click/tap on the “Submit feedback” button next to a module
5. The system will load up a “Submit feedback” form
6. The user will fill in the form
7. The system will submit the feedback

5. Alternative flows
5.1 System fails to retrieve data
The system shall display an error message.
The use case resumes at step 2.

6. Key Scenarios
6.1 System fails to retrieve data

7. Post-conditions
7.1 Successful completion
The user has submitted feedback for a module.
7.2 Failure condition
The system displays failure reasons accordingly.

Use case 2 (Feedback - Academic staff - View feedback for a module)

1. Brief description
This use case describes how the user will use Student Portal to view feedback for a module.

2. Actors
2.1 User (account type: academic staff)
2.2 System

3. Preconditions
3.1 The user has internet connection.
3.2 The user has an e-mail address.
3.3 The user has an internet browser installed.
3.4 The user has an account.
3.5 The user has signed into the system.

4. Basic flow of events
1. The system will load the "Overview" page
2. The user will click/tap on "Feedback"
3. The system will load up a list of feedback submissions
4. The user will click on the Feedback subject to view detailed information about the feedback submission

5. Alternative flows
5.1 System fails to retrieve data
The system shall display an error message.
The use case resumes at step 2.

6. Key Scenarios
6.1 System fails to retrieve data

7. Post-conditions
7.1 Successful completion
The user has viewed the feedback submission selected.
7.2 Failure condition
The system displays failure reasons accordingly.

Use case 3 (Feedback - Administrator - Approve feedback)

1. Brief description
This use case describes how the user will use Student Portal to approve submitted feedback.

2. Actors
2.1 User (account type: administrator)
2.2 System

3. Preconditions
3.1 The user has internet connection.
3.2 The user has an e-mail address.
3.3 The user has an internet browser installed.
3.4 The user has an account.
3.5 The user has signed into the system.

4. Basic flow of events
4.1 The system will load the "Overview" page
4.2 The user will click/tap on "Feedback"
4.3 The system will load up a list of feedback submissions which are pending approval
4.4 The user will click/tap the "Approve" button next to a feedback submission
4.5 The system will display a confirmation pop-up
4.6 The user will click/tap the "Approve" button
4.7 The system will approve the feedback submission

5. Alternative flows
5.1 System fails to retrieve data
The system shall display an error message.
The use case resumes at step 2.

6. Key Scenarios
6.1 System fails to retrieve data

7. Post-conditions
7.1 Successful completion
The user has approved the feedback selected.
7.2 Failure condition
The system displays failure reasons accordingly.

Use case 4 (Feedback - Administrator - Delete feedback)

1. Brief description
This use case describes how the user will use Student Portal to delete submitted feedback.

2. Actors
2.1 User (account type: administrator)
2.2 System

3. Preconditions
3.1 The user has internet connection.
3.2 The user has an e-mail address.
3.3 The user has an internet browser installed.
3.4 The user has an account.
3.5 The user has signed into the system.

4. Basic flow of events
4.1 The system will load the "Overview" page
4.2 The user will click/tap on "Feedback"
4.3 The system will load up a list of feedback submissions which are pending approval
4.4 The user will click/tap the "Delete" button next to a feedback submission
4.5 The system will display a confirmation pop-up
4.6 The user will click/tap the "Delete" button
4.7 The system will delete the feedback submission

5. Alternative flows
5.1 System fails to retrieve data
The system shall display an error message.
The use case resumes at step 2.

6. Key Scenarios
6.1 System fails to retrieve data

7. Post-conditions
7.1 Successful completion
The user has deleted the feedback selected.
7.2 Failure condition
The system displays failure reasons accordingly.

Messenger

Use case 1 (Messenger - Any account type - send a message to another user)

1. Brief description
This use case describes how the user will use Student Portal to send a message to another user.

2. Actors
2.1 User (account type: any)
2.2 System

3. Preconditions
3.1 The user has internet connection.
3.2 The user has an e-mail address.
3.3 The user has an internet browser installed.
3.4 The user has an account.
3.5 The user has signed into the system.

4. Basic flow of events
4.1 The system will load the "Home" page
4.2 The user will click/tap on "Messenger"
4.3 The system will load up a list of users
4.4 The user click/tap the "Message" button next to a user
4.5 The system will load up a "Send message" form
4.6 The user will fill in the form
4.7 The user will click/tap "Send message" button
4.8 The system will send the message

5. Alternative flows
5.1 System fails to retrieve data
The system shall display an error message.
The use case resumes at step 5.

6. Key Scenarios
6.1 System fails to retrieve data

7. Post-conditions
7.1 Successful completion
The user has successfully sent a message to another user.
7.2 Failure condition
The system displays failure reasons accordingly.

Use case 2 (Messenger - Any account type – Reply to a message)

1. Brief description
This use case describes how the user will use Student Portal to reply to a message.

2. Actors
2.1 User (account type: any)
2.2 System

3. Preconditions
3.1 The user has internet connection.
3.2 The user has an e-mail address.
3.3 The user has an internet browser installed.
3.4 The user has an account.
3.5 The user has signed into the system.

4. Basic flow of events
4.1 The system will load the "Home" page
4.2 The user will click/tap on "Messenger"
4.3 The system will load up a list of messages received by the user
4.4 The user will click/tap the "Reply" button next to a message
4.5 The system will load up a "Send message" form
4.6 The user will fill in the form
4.7 The user will click/tap "Send message" button
4.8 The system will send the message

5. Alternative flows
5.1 System fails to retrieve data
The system shall display an error message.
The use case resumes at step 5.

6. Key Scenarios
6.1 System fails to retrieve data

7. Post-conditions
7.1 Successful completion
The user has replied to a message.
7.2 Failure condition
The system displays failure reasons accordingly.

Use case 3 (Messenger - Any account type - Delete a message)

1. Brief description
This use case describes how the user will use Student Portal to delete a message.

2. Actors
2.1 User (account type: any)
2.2 System

3. Preconditions
3.1 The user has internet connection.
3.2 The user has an e-mail address.
3.3 The user has an internet browser installed.
3.4 The user has an account.
3.5 The user has signed into the system.

4. Basic flow of events
4.1 The system will load the “Home” page
4.2 The user will click/tap on “Messenger”
4.3 The system will load up a list of messages sent or received by the user
4.4 The user will click/tap the “Delete” button next to a message
4.5 The system will delete the message selected

5. Alternative flows
5.1 System fails to retrieve data
The system shall display an error message.
The use case resumes at step 5.

6. Key Scenarios
6.1 System fails to retrieve data

7. Post-conditions
7.1 Successful completion
The user has deleted the message selected.
7.2 Failure condition
The system displays failure reasons accordingly.

Account

Use case 1 (Account - Any account type - Update account)

1. Brief description
This use case describes how the user will use Student Portal to update their account.

2. Actors
2.1 User (account type: any)
2.2 System

3. Preconditions
3.1 The user has internet connection.
3.2 The user has an e-mail address.
3.3 The user has an internet browser installed.
3.4 The user has an account.
3.5 The user has signed into the system.

4. Basic flow of events
4.1 The system will load the “Home” page
4.2 The user will click/tap on “Account”
4.3 The user will click/tap on “Update account”
4.4 The system will load up an “Update account” form
4.5 The user will overwrite the form
4.6 The user will click/tap the “Update” button
4.7 The system will update the account

5. Alternative flows
None

6. Key Scenarios
None

7. Post-conditions
7.1 Successful completion
The user has updated their account.

7.2 Failure condition
None

Use case 2 (Account - Any account type - Change Password)

1. Brief description
This use case describes how the user will use Student Portal to change their password.

2. Actors
2.1 User (account type: any)
2.2 System

3. Preconditions
The user has internet connection.
The user has an e-mail address.
The user has an internet browser installed.
The user has an account.
The user has signed into the system.

4. Basic flow of events
1. The system will load the “Overview” page
2. The user will click/tap on “Account”
3. The user will click/tap on “Change Password”
4. The system will load up a “Change Password” form
5. The user will fill in the form
6. The user will click/tap the “Change password” button
7. The system will update the account’s password

5. Alternative flows
None

6. Key Scenarios
None

7. Post-conditions
7.1 Successful completion
The user has changed their password.
7.2 Failure condition
None

Use case 3 (Account - Pay course fees)

1. Brief description
This use case describes how the user will use Student Portal to pay their course fees.

2. Actors
2.1 User (account type: student)
2.2 System

3. Preconditions
3.1 The user has internet connection.
3.2 The user has an e-mail address.
3.3 The user has an internet browser installed.
3.4 The user has an account.
3.5 The user has signed into the system.

4. Basic flow of events
4.1 The system will load the “Overview” page
4.2 The user will click/tap on “Account” tile
4.3 The user will click/tap on “Pay course fees” tile
4.4 The system will load up a “Pay course fees” form
4.5 The user will fill in the form
4.6 The user will click/tap the “Pay with PayPal” button
4.7 The system will redirect to PayPal
4.8 The user will complete the payment on PayPal

5. Alternative flows
None

6. Key Scenarios
None

7. Post-conditions
7.1 Successful completion
The user has successfully paid for their course fees.

7.2 Failure condition
None

Use case 4 (Account - Any account type - Delete account)

1. Brief description
This use case describes how the user will use Student Portal to delete their account.

2. Actors
2.1 User (account type: any)
2.2 System

3. Preconditions
3.1 The user has internet connection.
3.2 The user has an e-mail address.
3.3 The user has an internet browser installed.
3.4 The user has an account.
3.5 The user has signed into the system.

4. Basic flow of events
4.1 The system will load the "Overview" page
4.2 The user will click/tap on "Account"
4.3 The user will click/tap on "Delete account"
4.4 The user will click/tap the "Delete account" button
4.5 The system will display a confirmation pop-up
4.6 The user will click/tap the "Delete" button
4.7 The system will delete the user's account.

5. Alternative flows
None

6. Key Scenarios
None

7. Post-conditions
7.1 Successful completion
The user has deleted their account.
7.2 Failure condition
None

Use case 5 (Account - Administrator - Create an account)

1. Brief description
This use case describes how the user will use Student Portal to create an account.

2. Actors
2.1 User (account type: administrator)
2.2 System

3. Preconditions
3.1 The user has internet connection.
3.2 The user has an e-mail address.
3.3 The user has an internet browser installed.
3.4 The user has an account.
3.5 The user has signed into the system.

4. Basic flow of events
4.1. The system will load the "Home" page
4.2 The user will click/tap on "Account"
4.3 The user will click/tap the "Create account" button
4.4 The system will load up a "Create account" form
4.5 The user will fill in the form
4.6 The user will click/tap the "Create account" button
4.7 The system will create the account

5. Alternative flows
None

6. Key Scenarios
None

7. Post-conditions
7.1 Successful completion
The user has created an account.
7.2 Failure condition
None

Use case 6 (Account - Administrator - Update an account)

1. Brief description
This use case describes how the user will use Student Portal to update an account.

2. Actors
2.1 User (account type: administrator)
2.2 System

3. Preconditions
3.1 The user has internet connection.
3.2 The user has an e-mail address.
3.3 The user has an internet browser installed.
3.4 The user has an account.
3.5 The user has signed into the system.

4. Basic flow of events
4.1 The system will load the “Overview” page
4.2 The user will click/tap on “Account”
4.3 The system will load up a list of already existing accounts
4.4 The user will click/tap the “Update account” button next to an account
4.5 The system will load up the “Update account” form
4.6 The user will overwrite the form
4.7 The user will click/tap the “Update account” button
4.8 The system will update the account selected

5. Alternative flows
None

6. Key Scenarios
None

7. Post-conditions
7.1 Successful completion
The user has updated the account selected.
7.2 Failure condition
None

Use case 7 (Account - Administrator - Deactivate an account)

1. Brief description
This use case describes how the user will use Student Portal to deactivate an account.

2. Actors
2.1 User (account type: administrator)
2.2 System

3. Preconditions
3.1 The user has internet connection.
3.2 The user has an e-mail address.
3.3 The user has an internet browser installed.
3.4 The user has an account.
3.5 The user has signed into the system.

4. Basic flow of events
4.1 The system will load the "Home" page
4.2 The user will click/tap on "Account"
4.3 The system will load up a list of already existing accounts
4.4 The user will click/tap the "account" button next to an account
4.5 The system will deactivate the account selected.

5. Alternative flows
5.1 System fails to retrieve data
The system shall display an error message.
The use case resumes at step 4.2.

6. Key Scenarios
6.1 System fails to retrieve data

7. Post-conditions
7.1 Successful completion
The user has successfully deactivated the account selected.
7.2 Failure condition
The system displays failure reasons accordingly.

Use case 8 (Account - Administrator - Reactivate an account)

1. Brief description
This use case describes how the user will use Student Portal to reactivate an account.

2. Actors
2.1 User (account type: administrator)
2.2 System

3. Preconditions
3.1 The user has internet connection.
3.2 The user has an e-mail address.
3.3 The user has an internet browser installed.
3.4 The user has an account.
3.5 The user has signed into the system.

4. Basic flow of events
4.1 The system will load the "Home" page
4.2 The user will click/tap on "Account"
4.3 The system will load up a list of already existing accounts
4.4 The user will click/tap the "Reactivate" button next to an account
4.5 The system archive the account selected

5. Alternative flows
5.1 System fails to retrieve data
The system shall display an error message.
The use case resumes at step 2.

6. Key Scenarios
6.1 System fails to retrieve data

7. Post-conditions
7.1 Successful completion
The user has archived the account selected.
7.2 Failure condition
The system displays failure reasons accordingly.

Use case 9 (Account - Administrator - Delete an account)

1. Brief description
This use case describes how the user will use Student Portal to delete an account.

2. Actors
2.1 User (account type: any)
2.2 System

3. Preconditions
3.1 The user has internet connection.
3.2 The user has an e-mail address.
3.3 The user has an internet browser installed.
3.4 The user has an account.
3.5 The user has signed into the system.

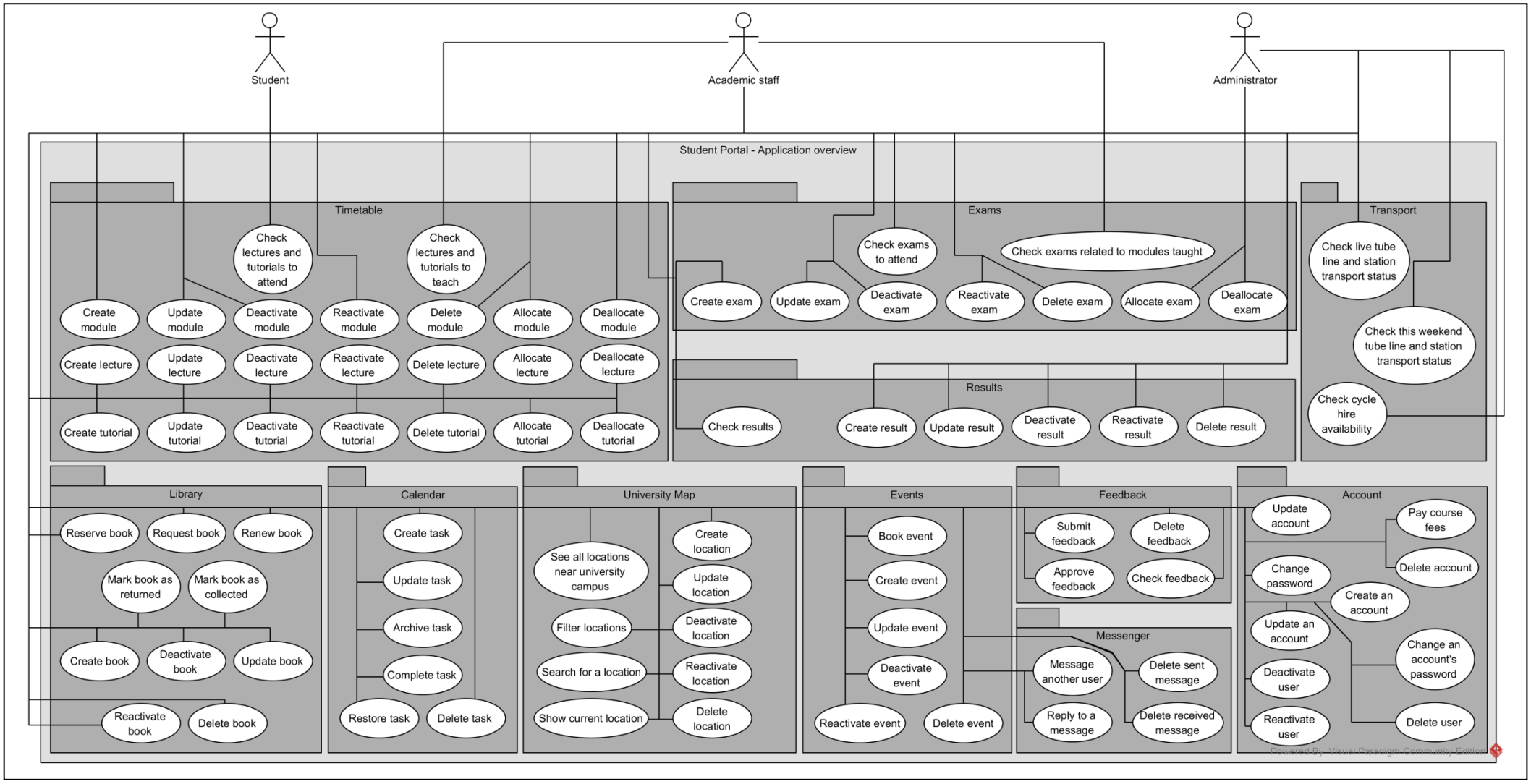
4. Basic flow of events
4.1 The system will load the "Home" page
4.2 The user will click/tap on "Account"
4.3 The system will load up a list of already existing accounts
4.4 The user will click/tap the "Delete" button next to an account
4.5 The system will delete the account selected.

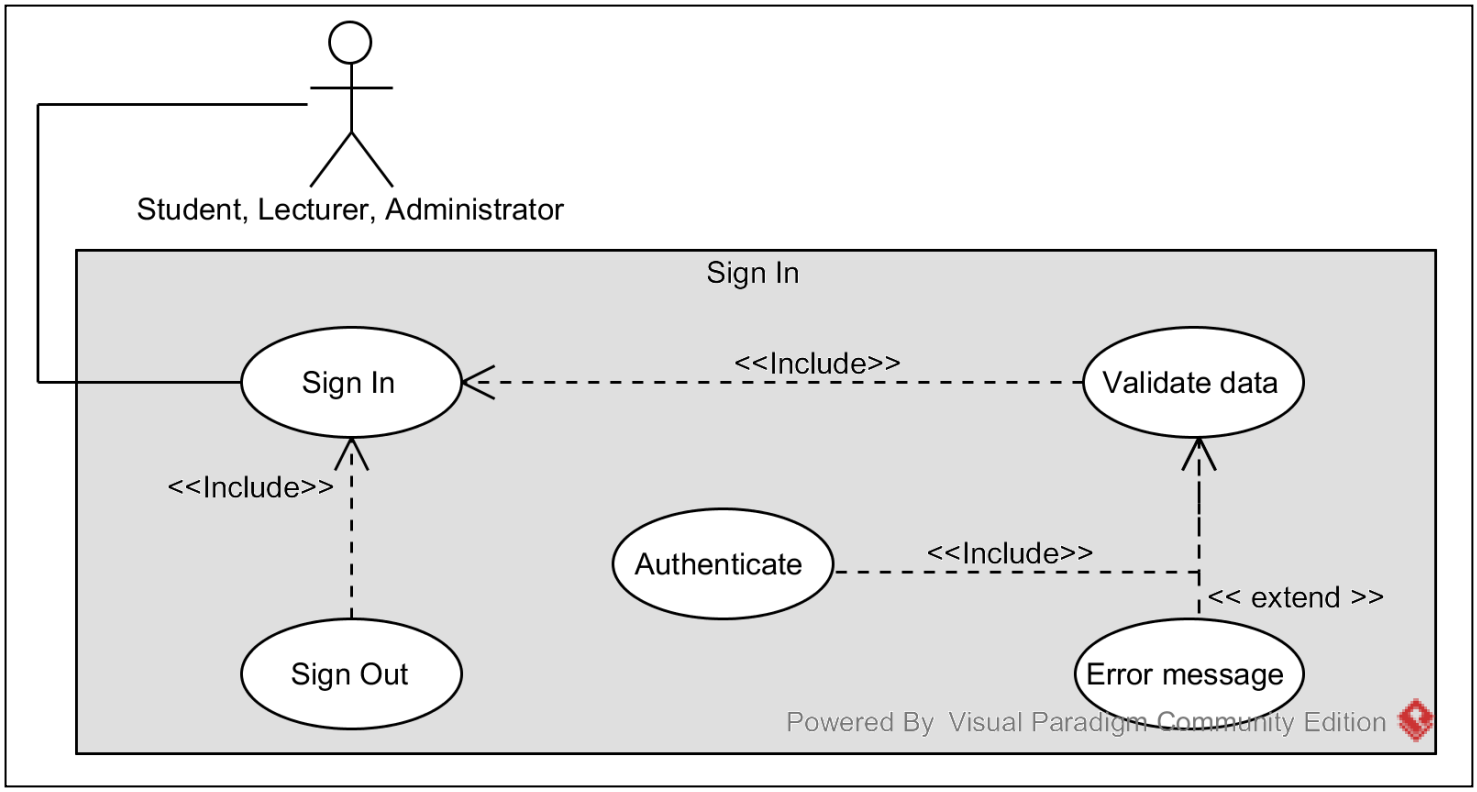
5. Alternative flows
5.1 System fails to retrieve data
The system shall display an error message.
The use case resumes at step 4.2.

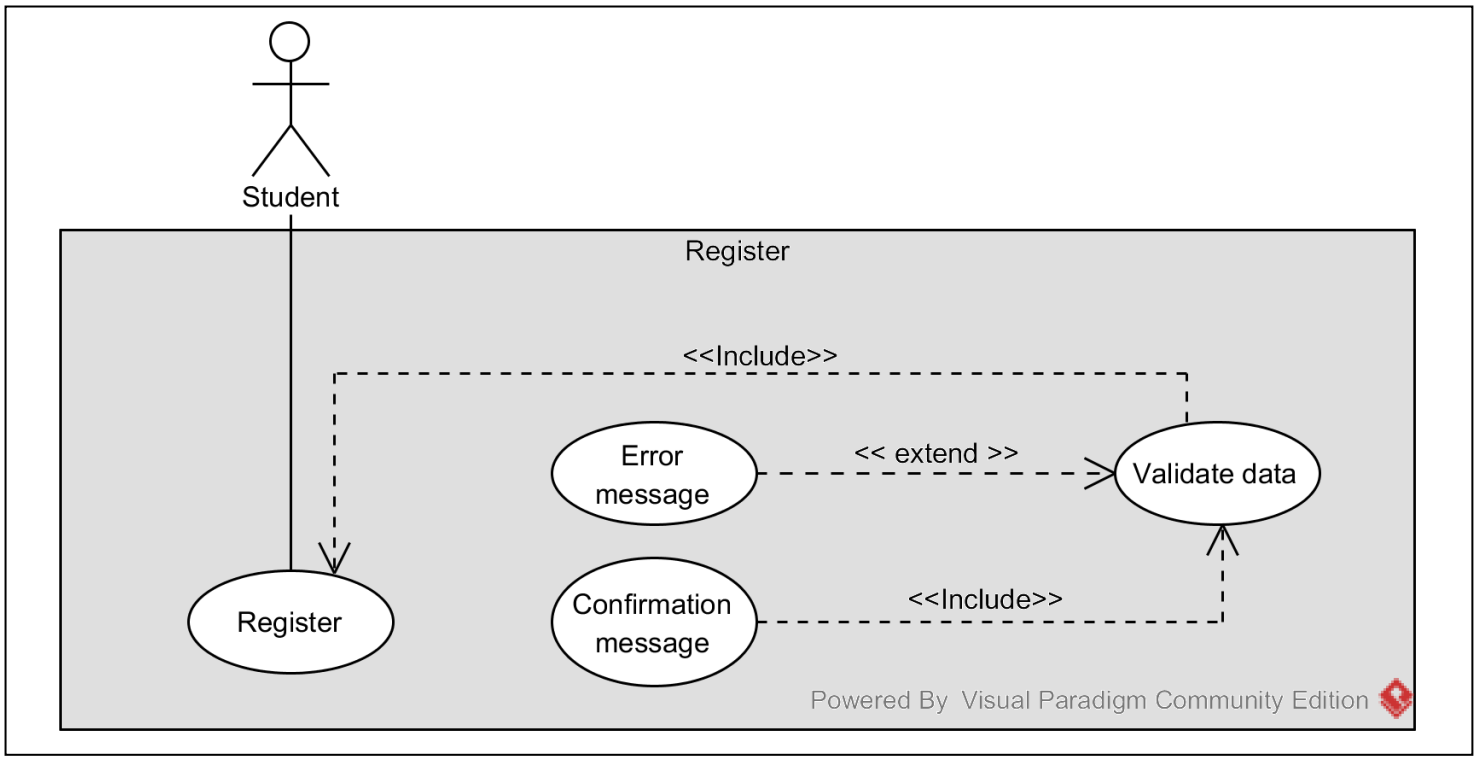
6. Key Scenarios
6.1 System fails to retrieve data

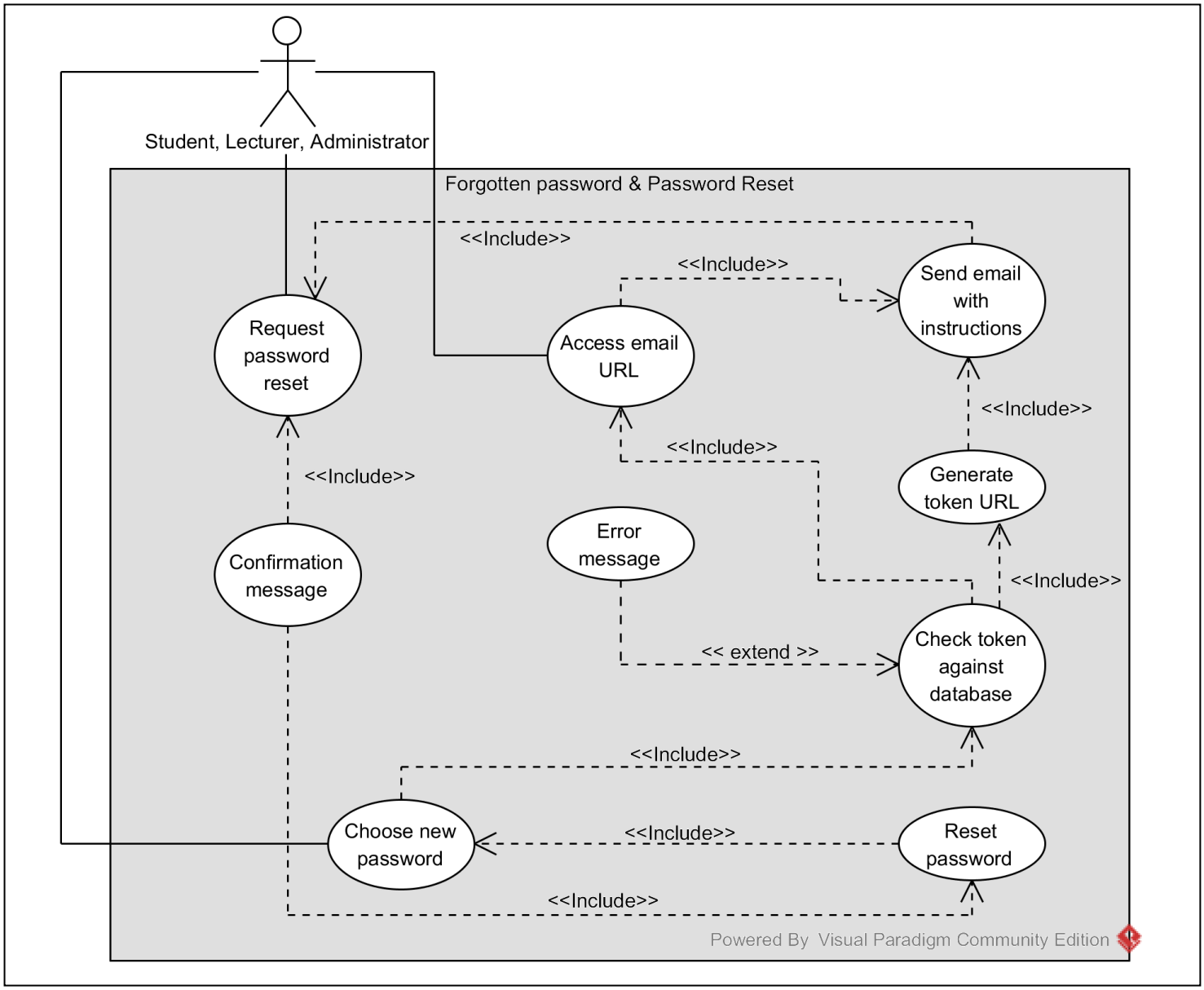
7. Post-conditions
7.1 Successful completion
The user has deleted the account selected.
7.2 Failure condition
The system displays failure reasons accordingly.

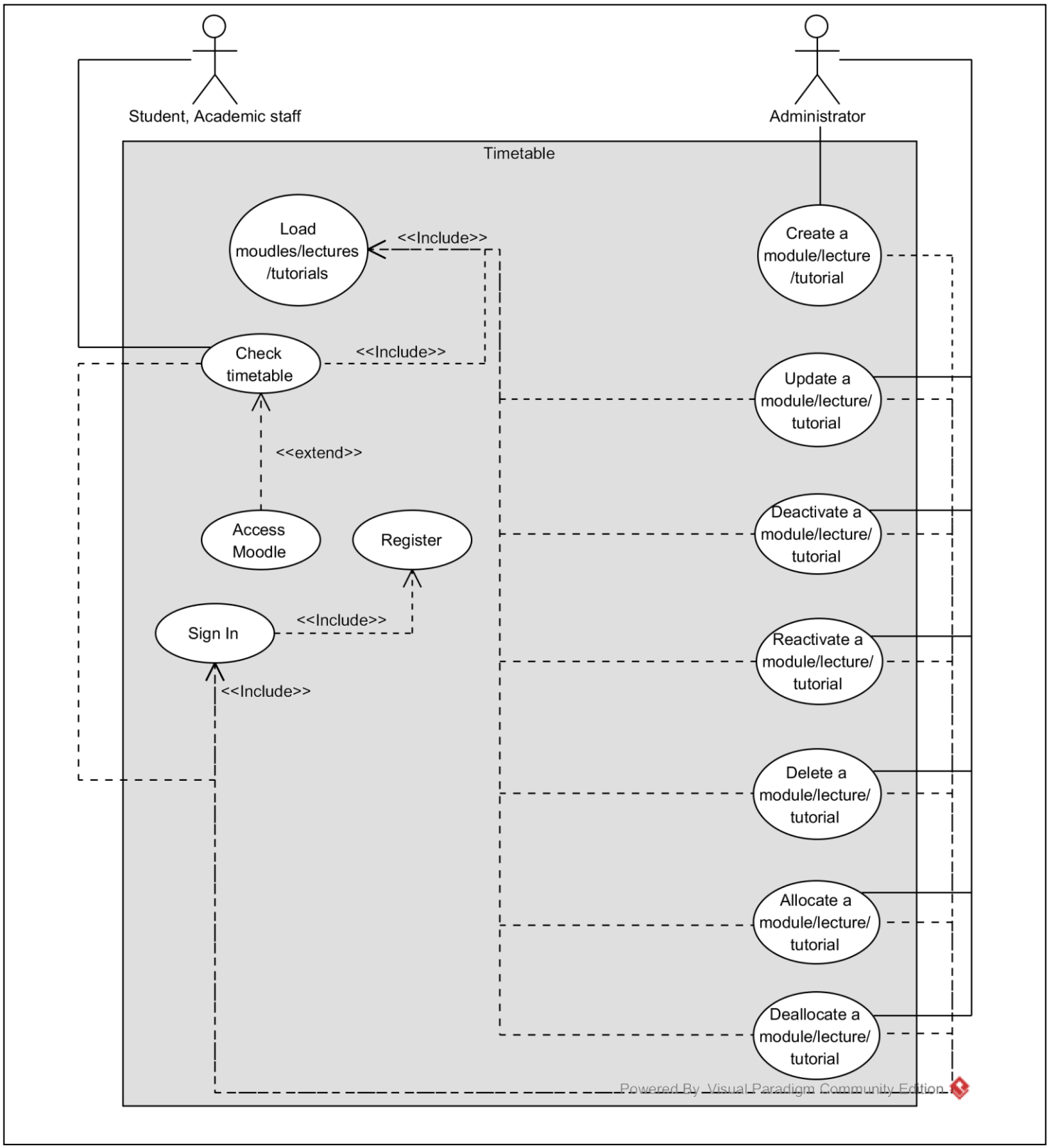
2. Use case diagrams

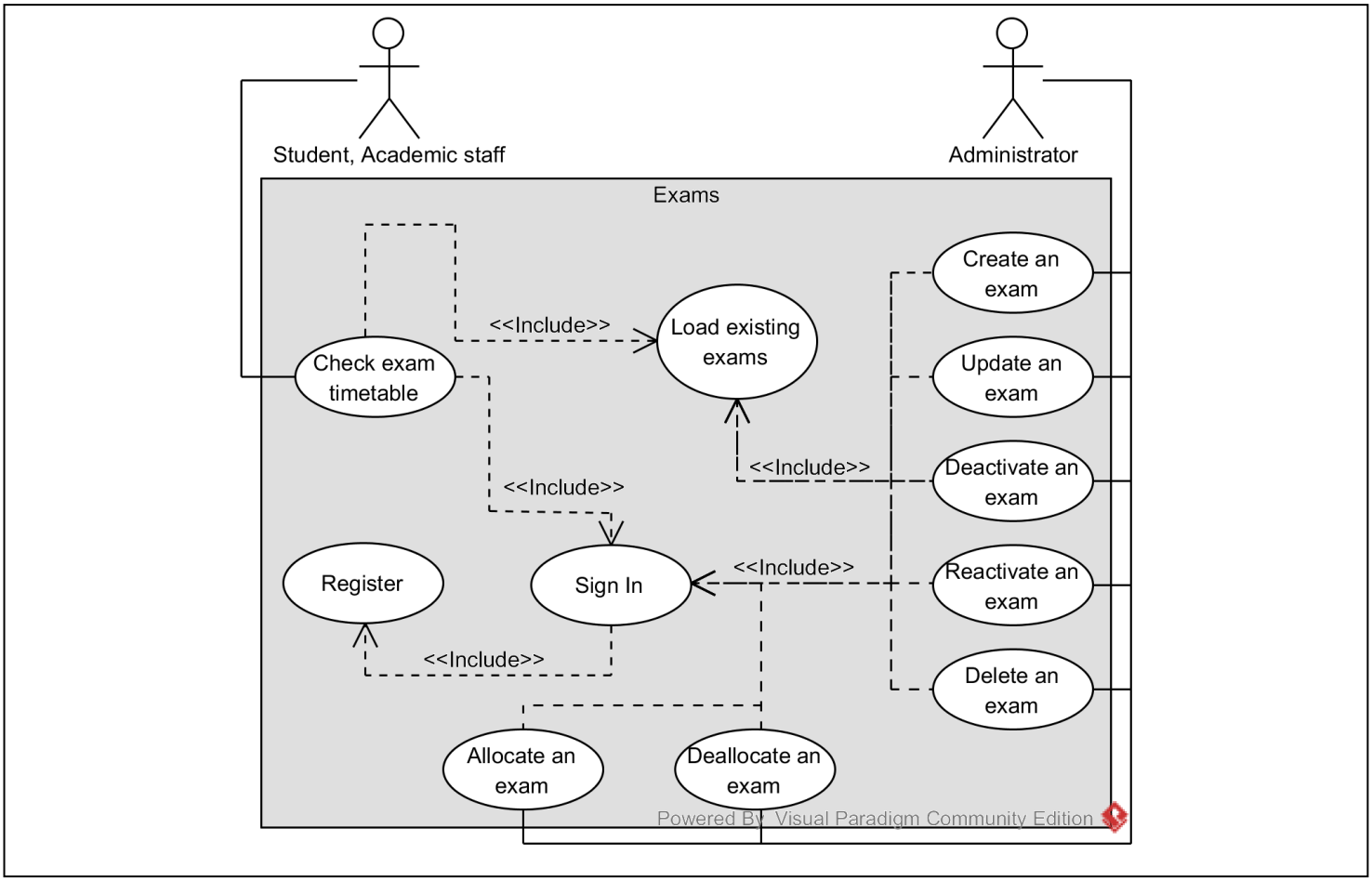


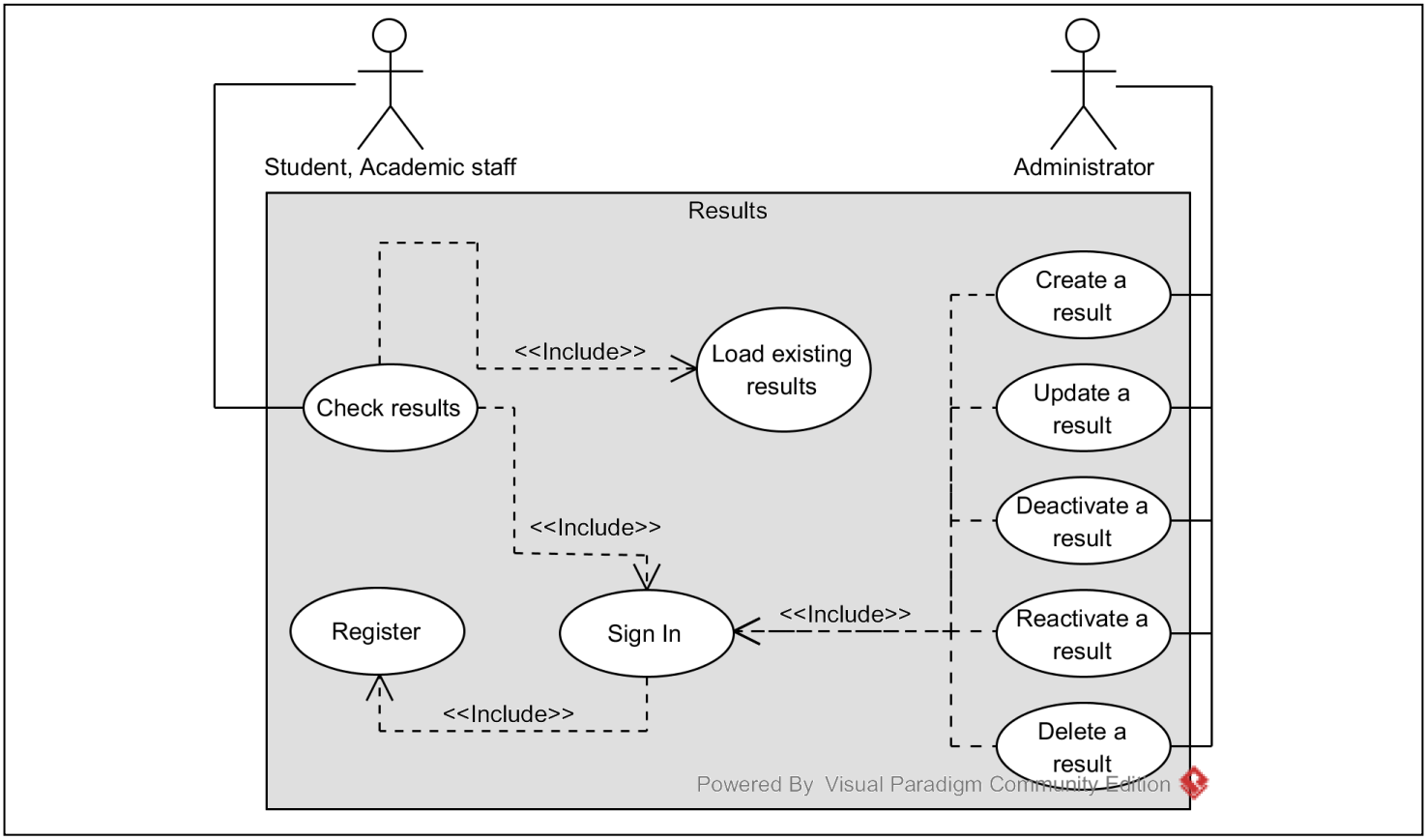

Student, Lecturer, Administrator
Sign In
Sign In
<<Include>>
Validate data
<<Include>>
Authenticate
<<Include>>
<< extend >>
Sign Out
Error message
Powered By Visual Paradigm Community Edition

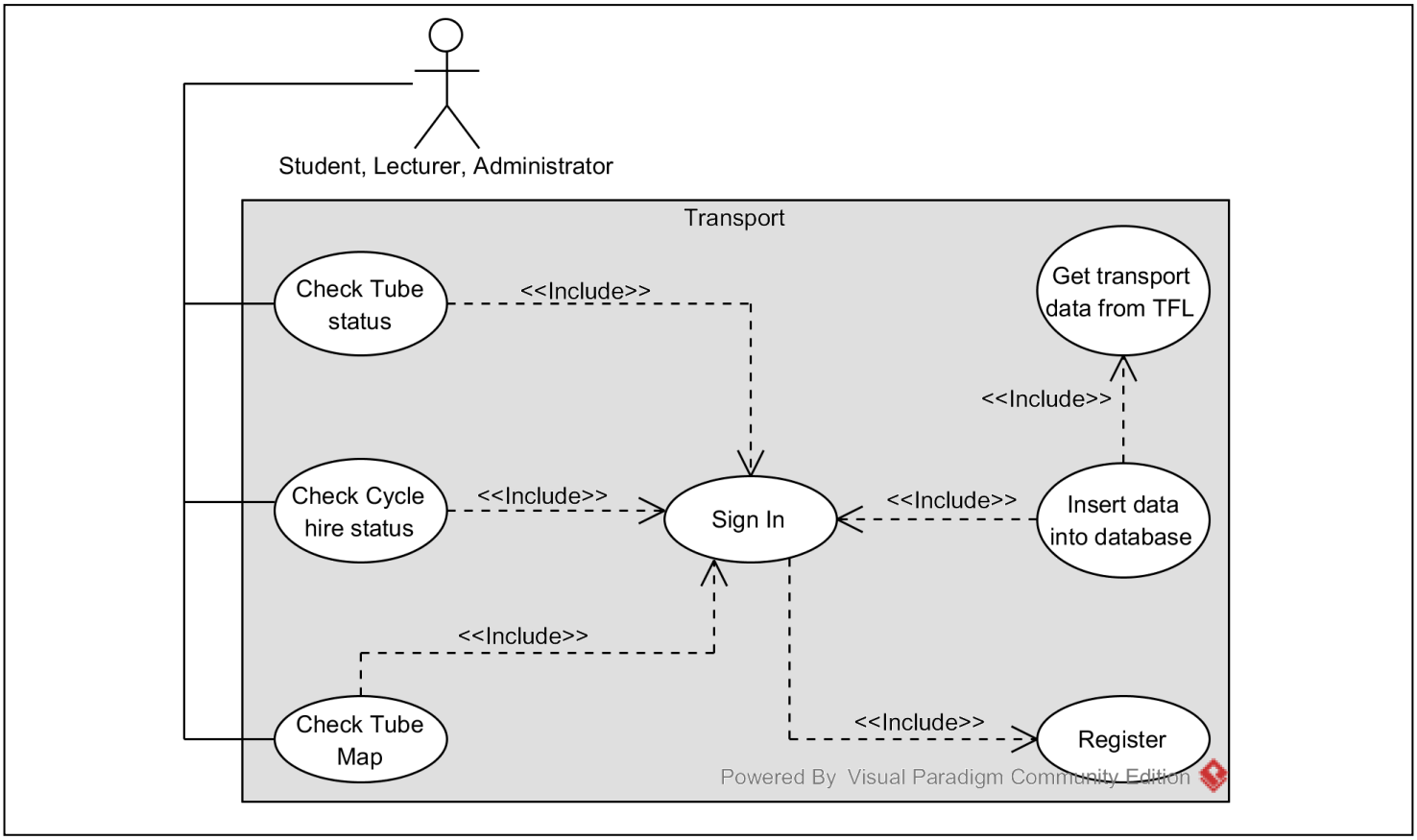

Student
Register
<<Include>>
Error message
<< extend >>
Validate data
Register
Confirmation message
<<Include>>
Powered By Visual Paradigm Community Edition

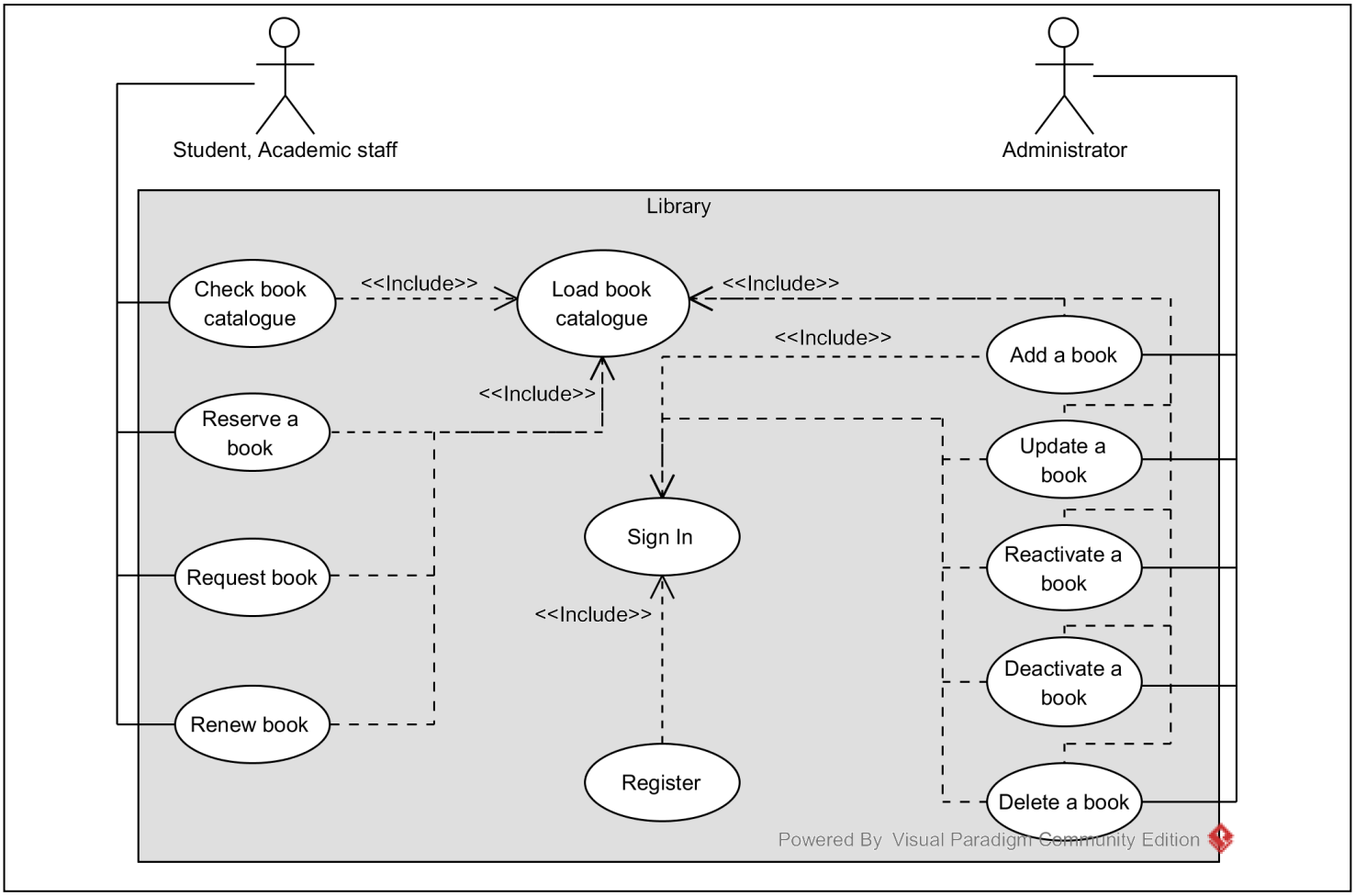

Student, Lecturer, Administrator
Forgotten password & Password Reset
<<Include>>
<<Include>>
Send email with instructions
Request password reset
Access email URL
<<Include>>
<<Include>>
<<Include>>
Generate token URL
Error message
Confirmation message
<<Include>>
Check token against database
<< extend >>
<<Include>>
Choose new password
<<Include>>
Reset password
<<Include>>
Powered By Visual Paradigm Community Edition

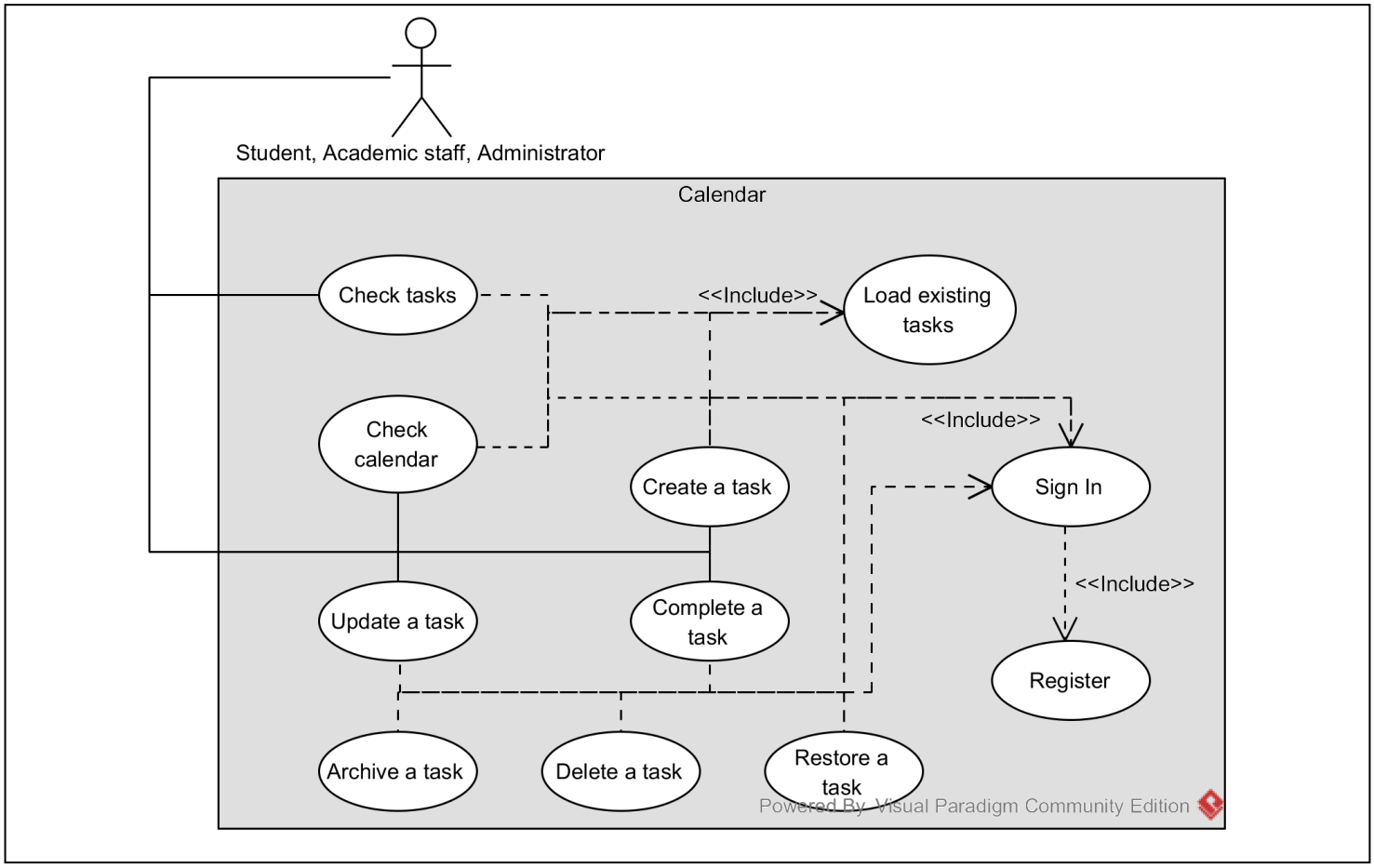

Student, Academic staff
Administrator
Timetable
Load moudles/lectures /tutorials
<<Include>>
Create a module/lecture /tutorial
Check timetable
<<Include>>
Update a module/lecture/ tutorial
<<extend>>
Access Moodle
Register
Deactivate a module/lecture/ tutorial
Sign In
<<Include>>
Reactivate a module/lecture/ tutorial
<<Include>>
Delete a module/lecture/ tutorial
Allocate a module/lecture/ tutorial
Deallocate a module/lecture/ tutorial
Powered By Visual Paradigm Community Edition

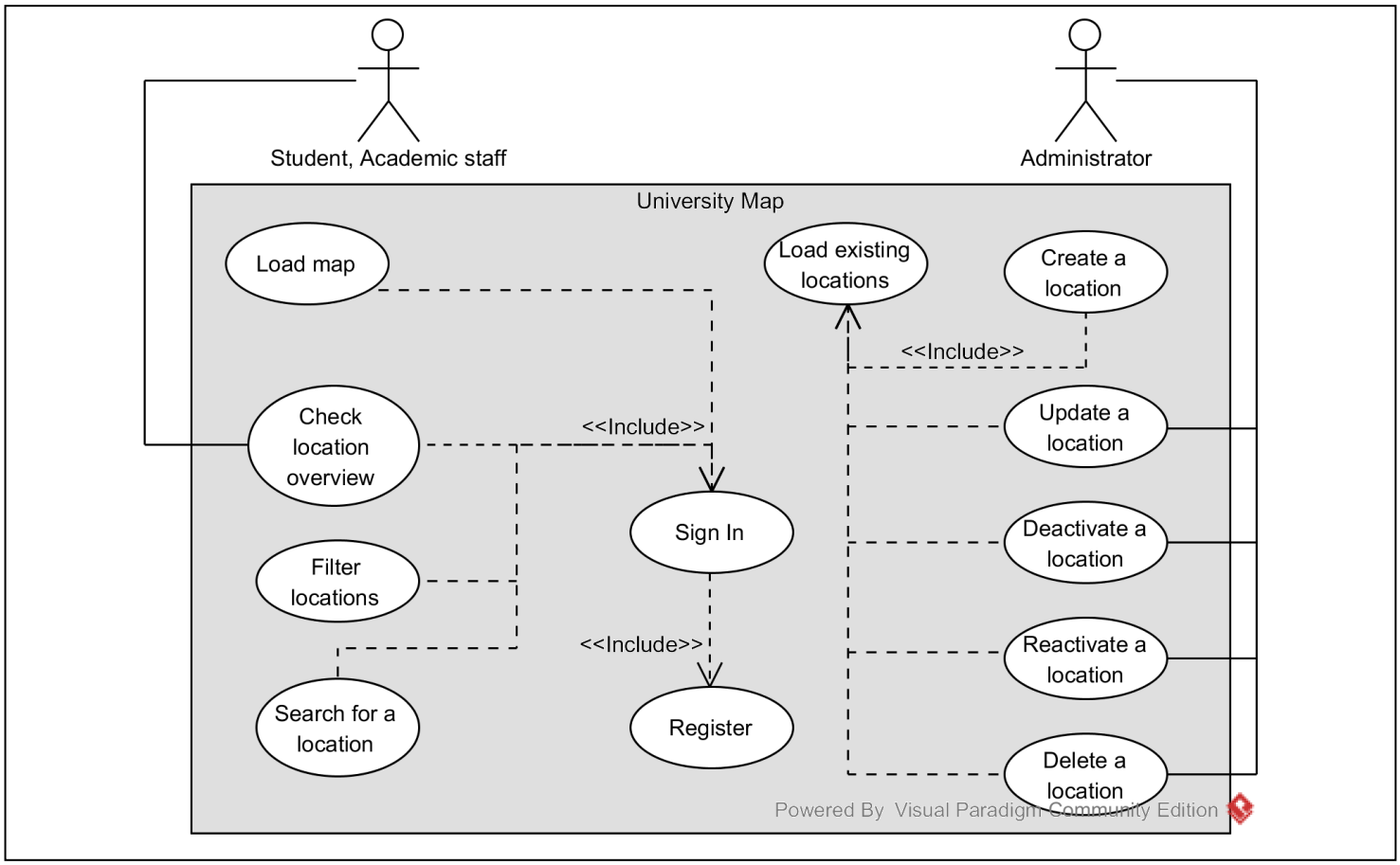

Student, Academic staff
Administrator
Exams
Create an exam
Load existing exams
<<Include>>
Check exam timetable
Update an exam
<<Include>>
Deactivate an exam
<<Include>>
Register
Sign In
<<Include>>
Reactivate an exam
<<Include>>
Delete an exam
Allocate an exam
Deallocate an exam
Powered By Visual Paradigm Community Edition

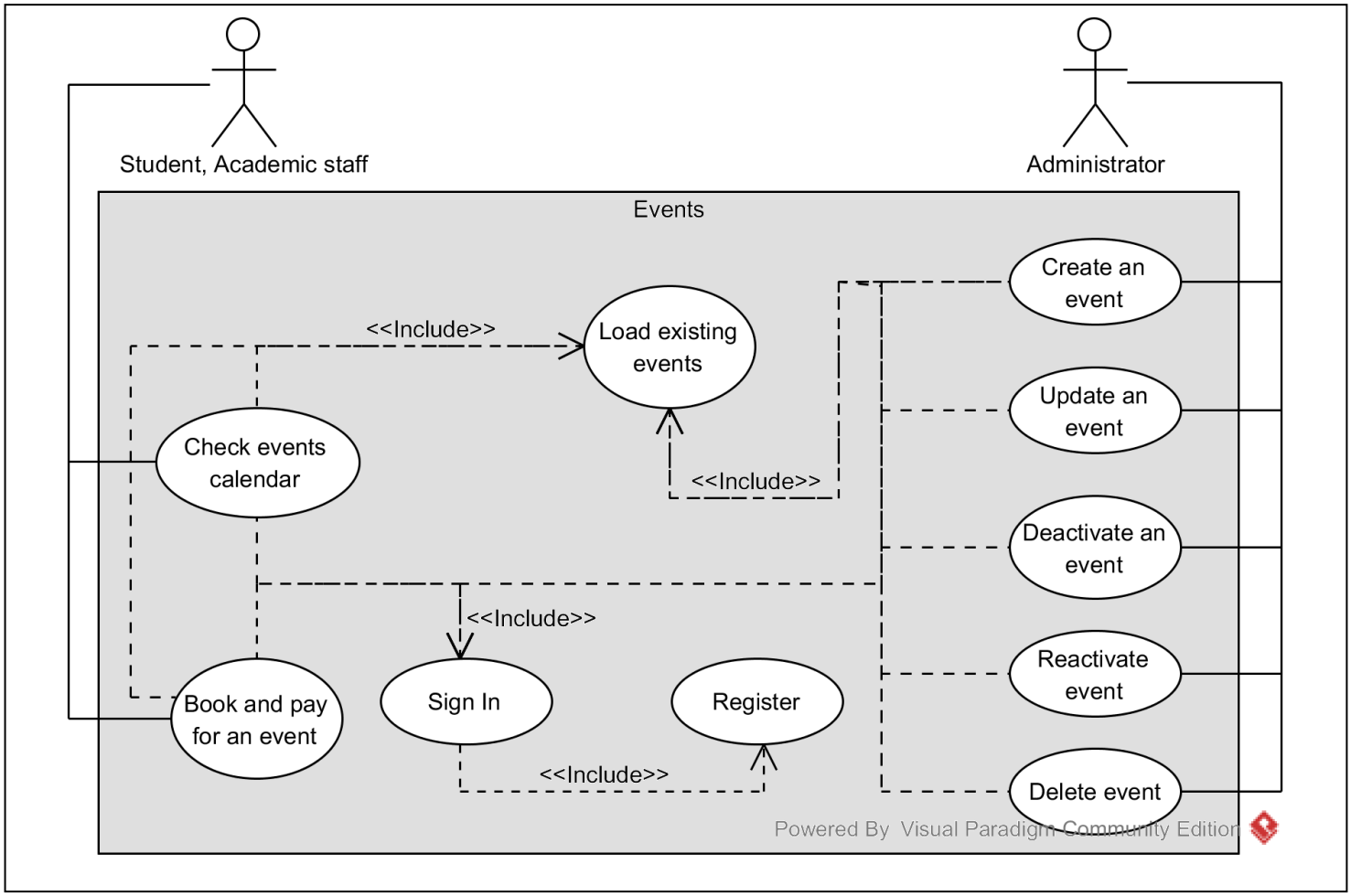

Student, Academic staff
Administrator
Results
Create a result
Load existing results
<<Include>>
Check results
Update a result
<<Include>>
Deactivate a result
Register
Sign In
<<Include>>
Reactivate a result
<<Include>>
Delete a result
Powered By Visual Paradigm Community Edition

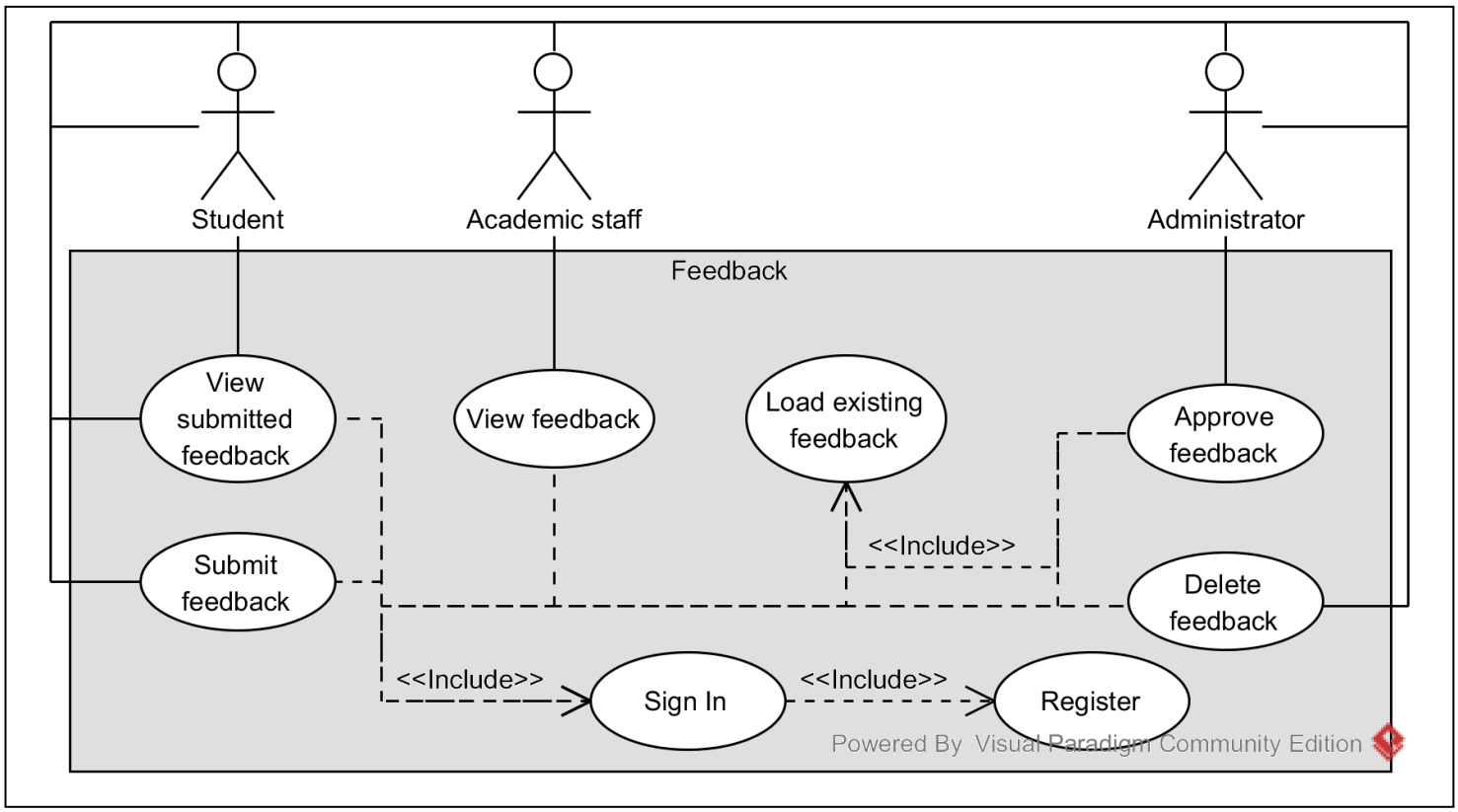

Student, Lecturer, Administrator
Transport
Check Tube status
<<Include>>
Get transport data from TFL
<<Include>>
Check Cycle hire status
<<Include>>
Sign In
<<Include>>
Insert data into database
<<Include>>
Check Tube Map
<<Include>>
Register
Powered By Visual Paradigm Community Edition

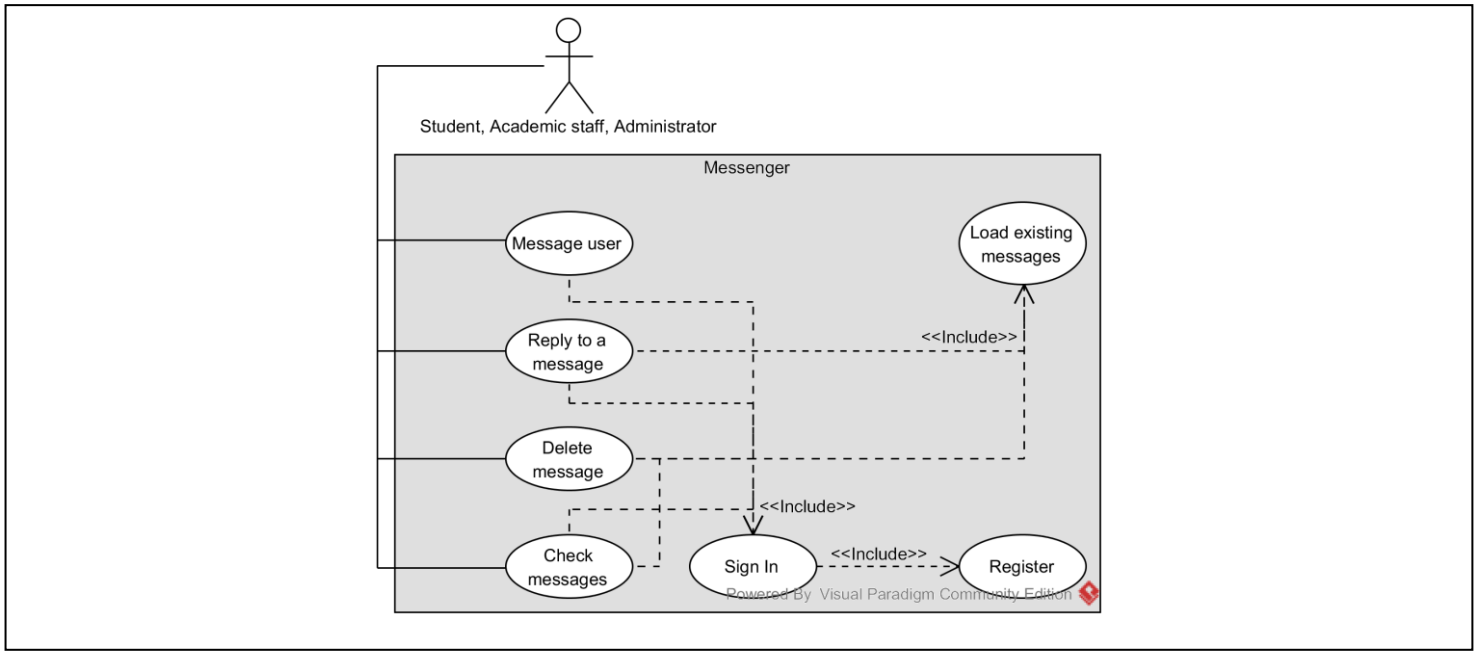

Student, Academic staff
Administrator
Library
Check book catalogue
<<Include>>
Load book catalogue
<<Include>>
<<Include>>
Add a book
<<Include>>
Reserve a book
Update a book
Sign In
Request book
Reactivate a book
<<Include>>
Deactivate a book
Renew book
Register
Delete a book
Powered By Visual Paradigm Community Edition

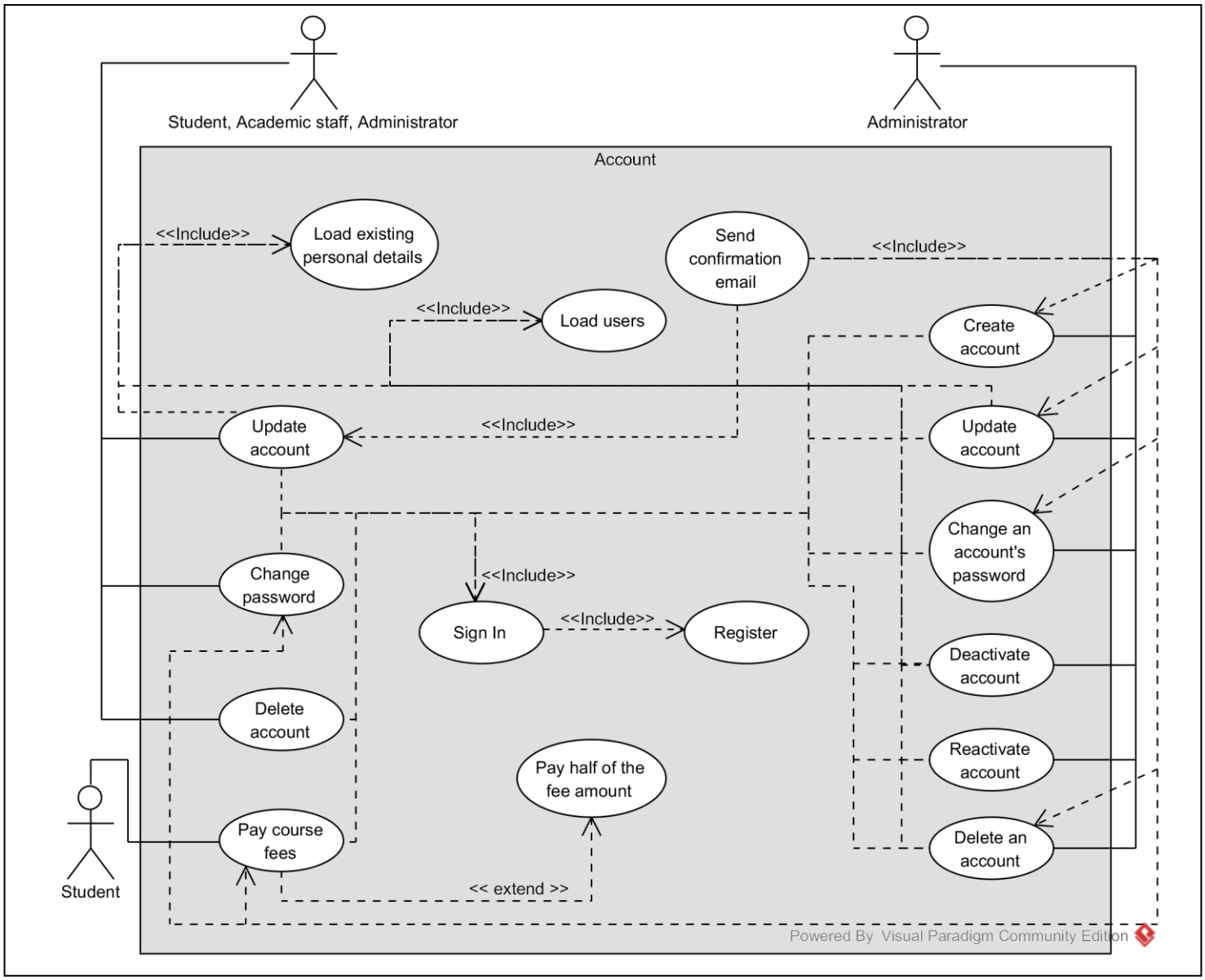

Student, Academic staff, Administrator
Calendar
Check tasks
<<Include>>
Load existing tasks
Check calendar
<<Include>>
Create a task
Sign In
<<Include>>
Update a task
Complete a task
Register
Archive a task
Delete a task
Restore a task
Powered By Visual Paradigm Community Edition


Student, Academic staff
Administrator
University Map
Load map
Load existing locations
Create a location
<<Include>>
Check location overview
<<Include>>
Update a location
Sign In
Deactivate a location
Filter locations
<<Include>>
Reactivate a location
Search for a location
Register
Delete a location
Powered By Visual Paradigm Community Edition

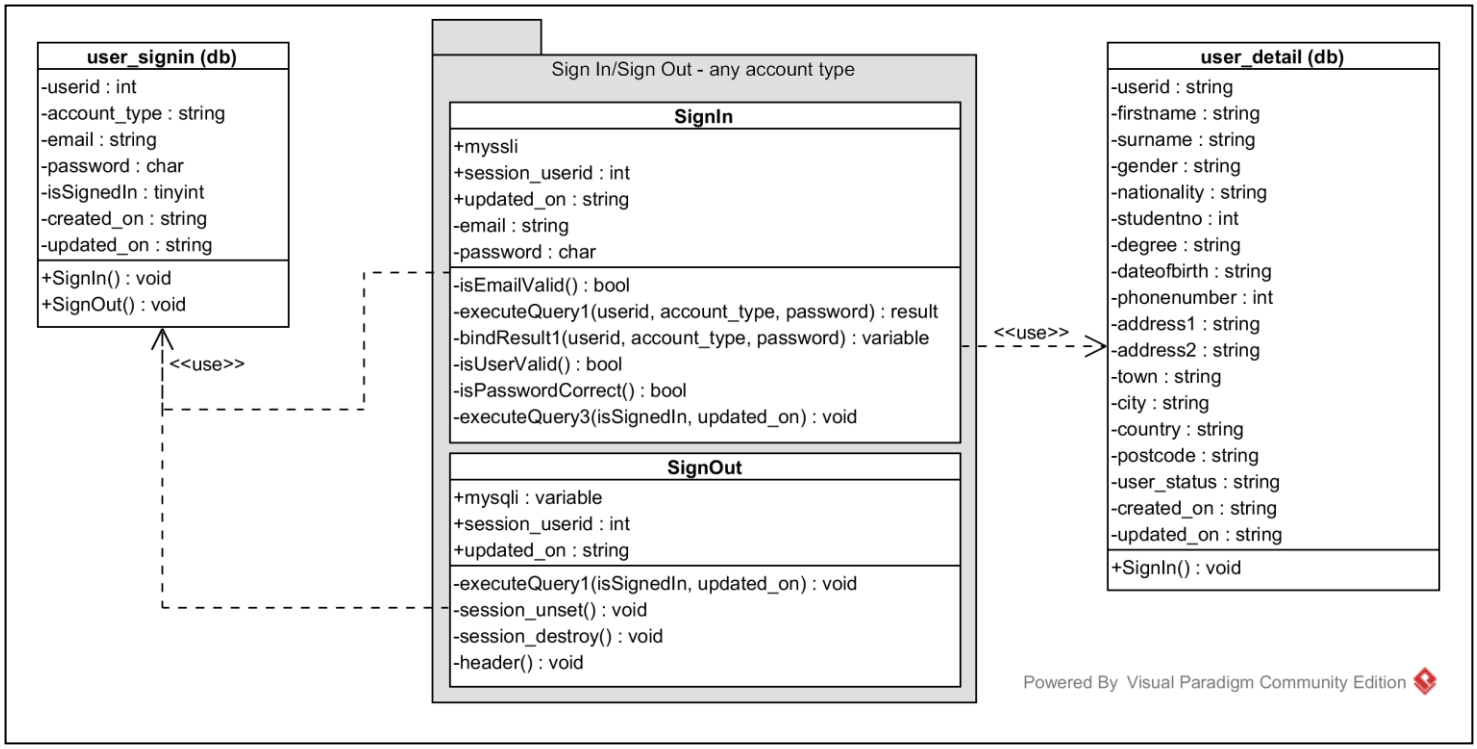

Student, Academic staff
Administrator
Events
Create an event
<<Include>>
Load existing events
Update an event
Check events calendar
<<Include>>
Deactivate an event
<<Include>>
Reactivate event
Book and pay for an event
Sign In
Register
<<Include>>
Delete event
Powered By Visual Paradigm Community Edition

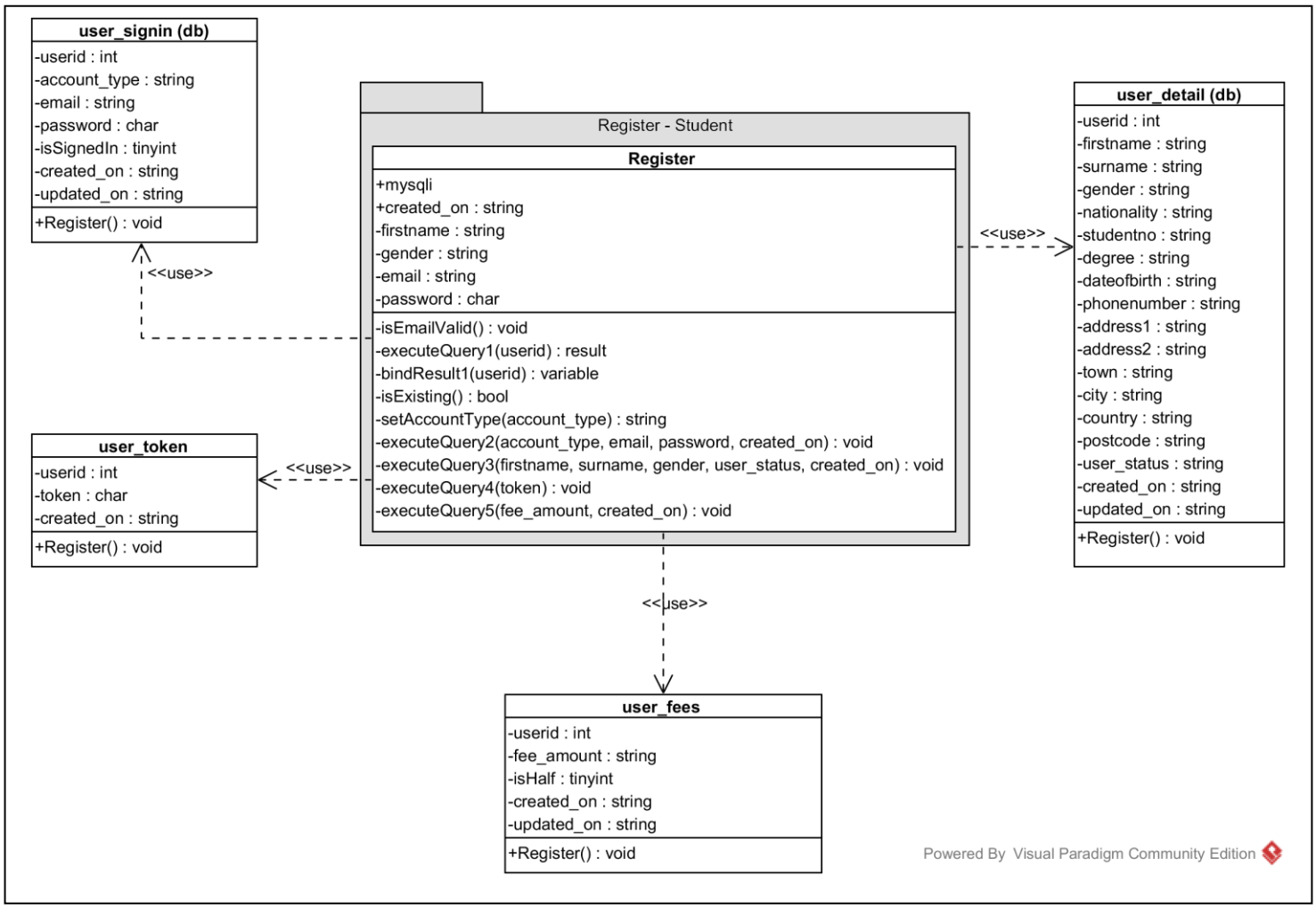

Student
Academic staff
Administrator
Feedback
View submitted feedback
View feedback
Load existing feedback
Approve feedback
<<Include>>
Submit feedback
Delete feedback
<<Include>>
Sign In
<<Include>>
Register
Powered By Visual Paradigm Community Edition

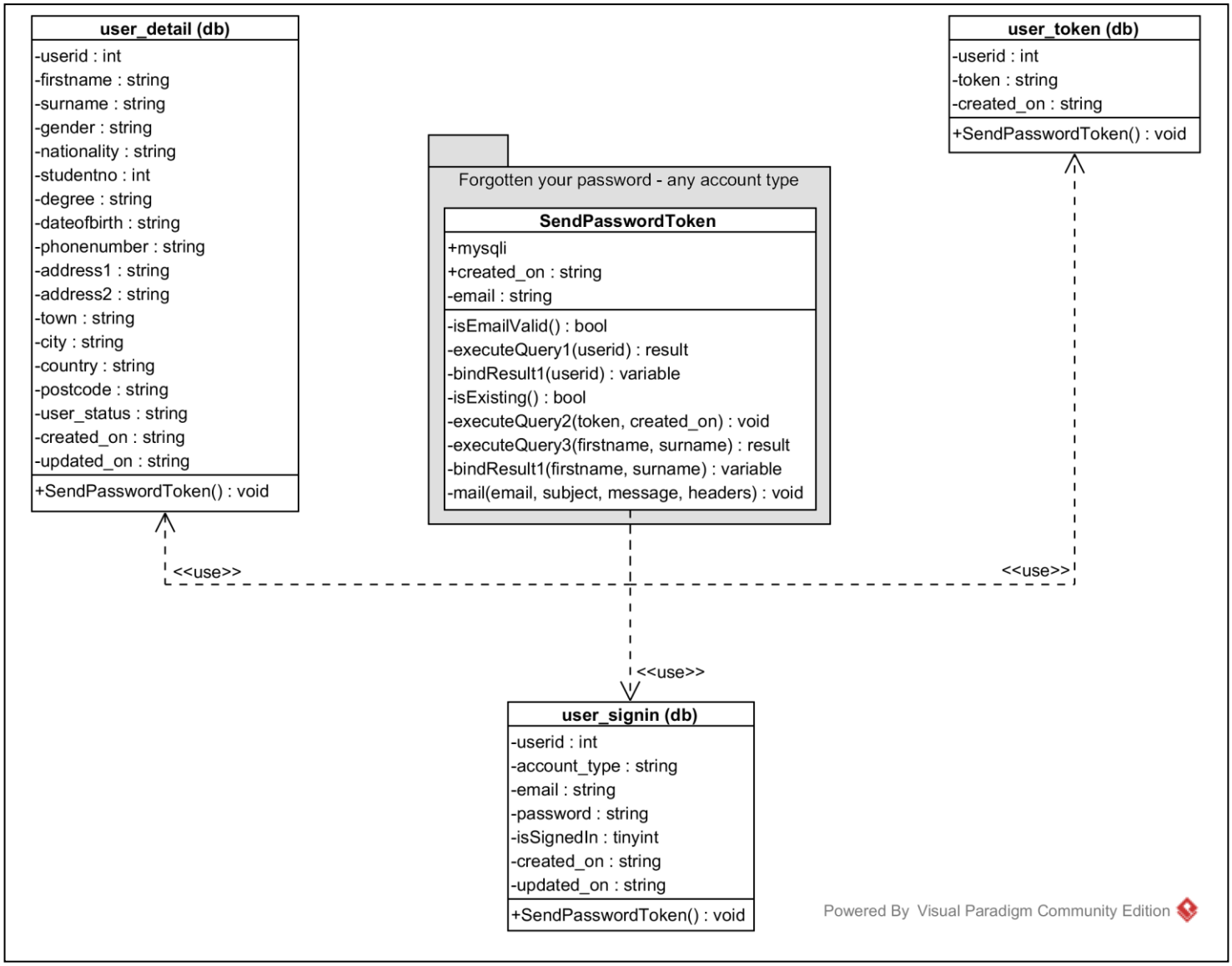

Student, Academic staff, Administrator
Messenger
Message user
Load existing messages
Reply to a message
<<Include>>
Delete message
<<Include>>
Check messages
Sign In
<<Include>>
Register
Powered By Visual Paradigm Community Edition

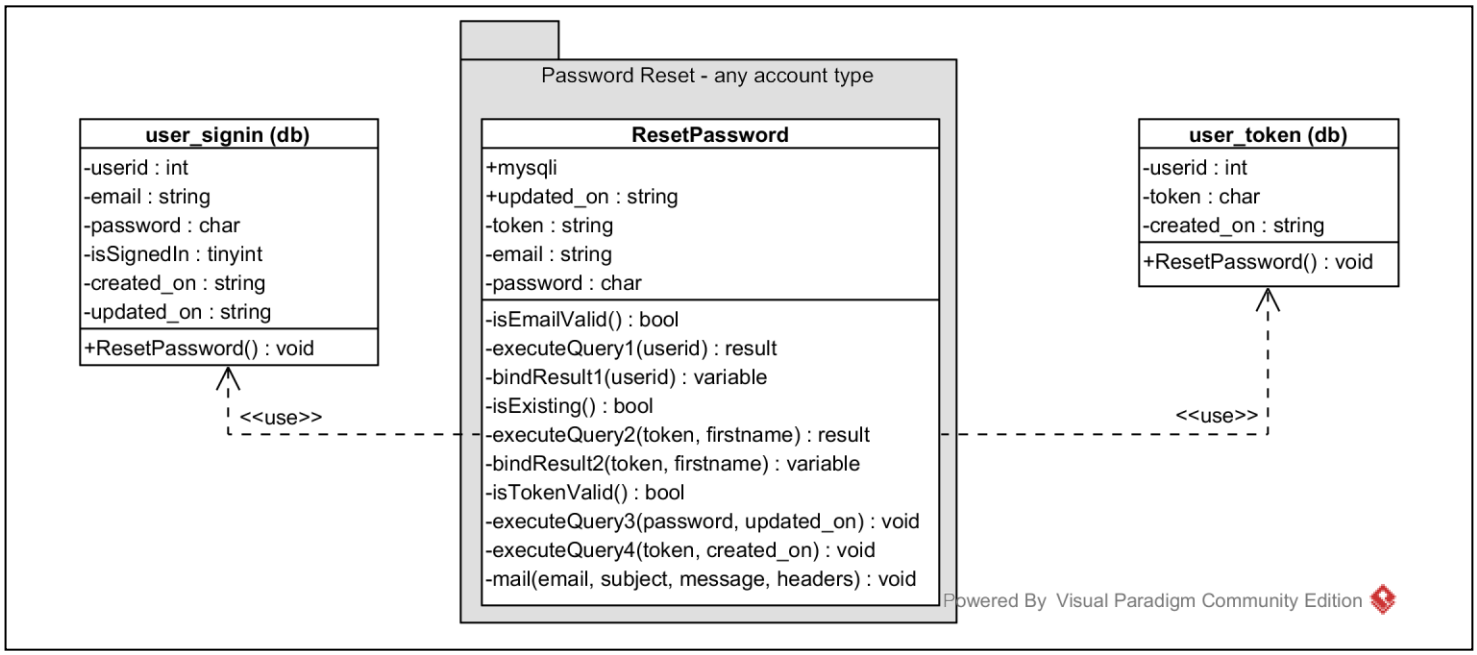

Student, Academic staff, Administrator
Administrator
Account
<<Include>>
Load existing personal details
Send confirmation email
<<Include>>
<<Include>>
Load users
Create account
Update account
<<Include>>
Update account
Change an account's password
Change password
<<Include>>
Sign In
<<Include>>
Register
Deactivate account
Delete account
Reactivate account
Pay half of the fee amount
Pay course fees
Delete an account
Student
<< extend >>
Powered By Visual Paradigm Community Edition

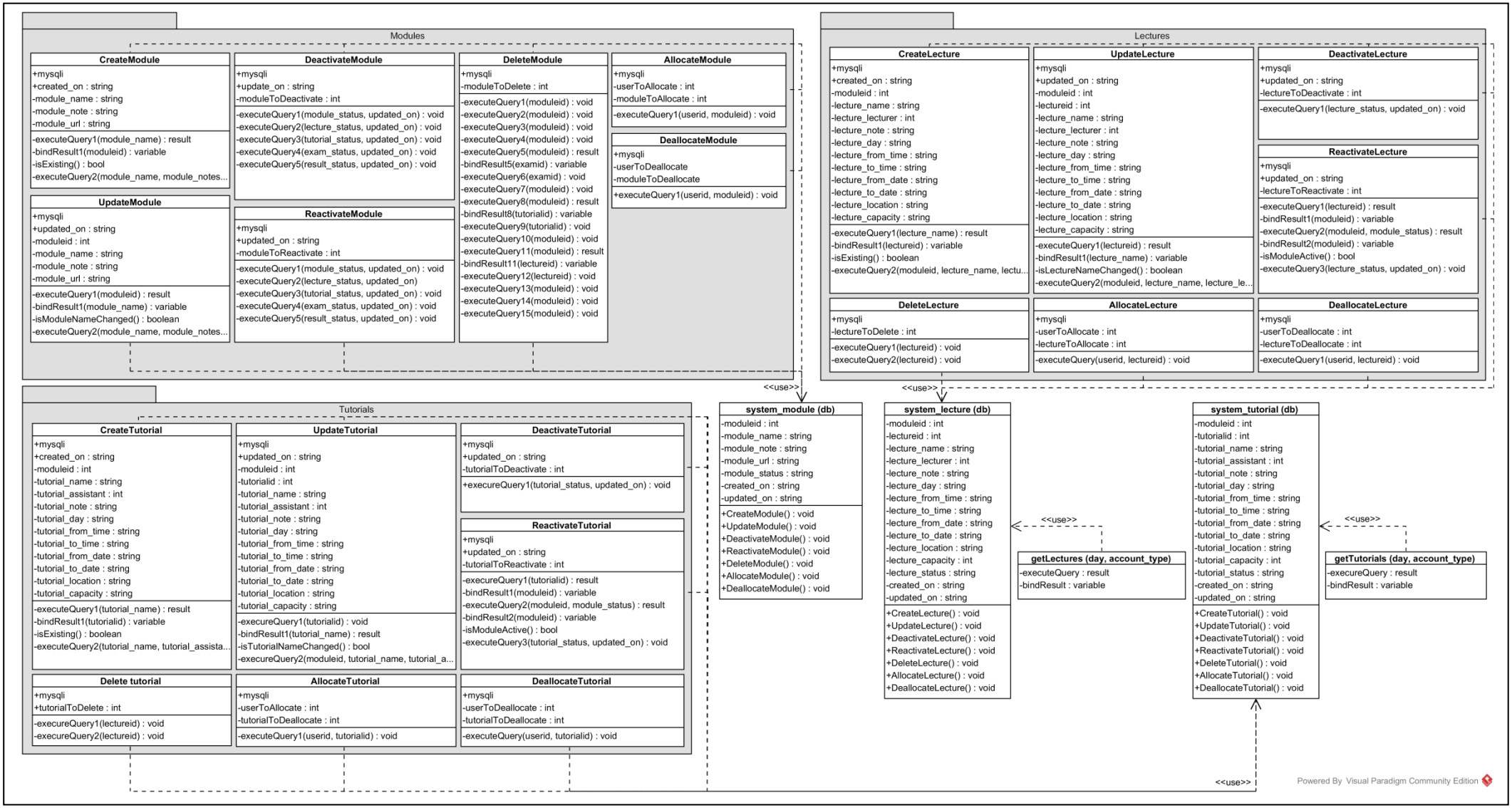
3. Class diagrams

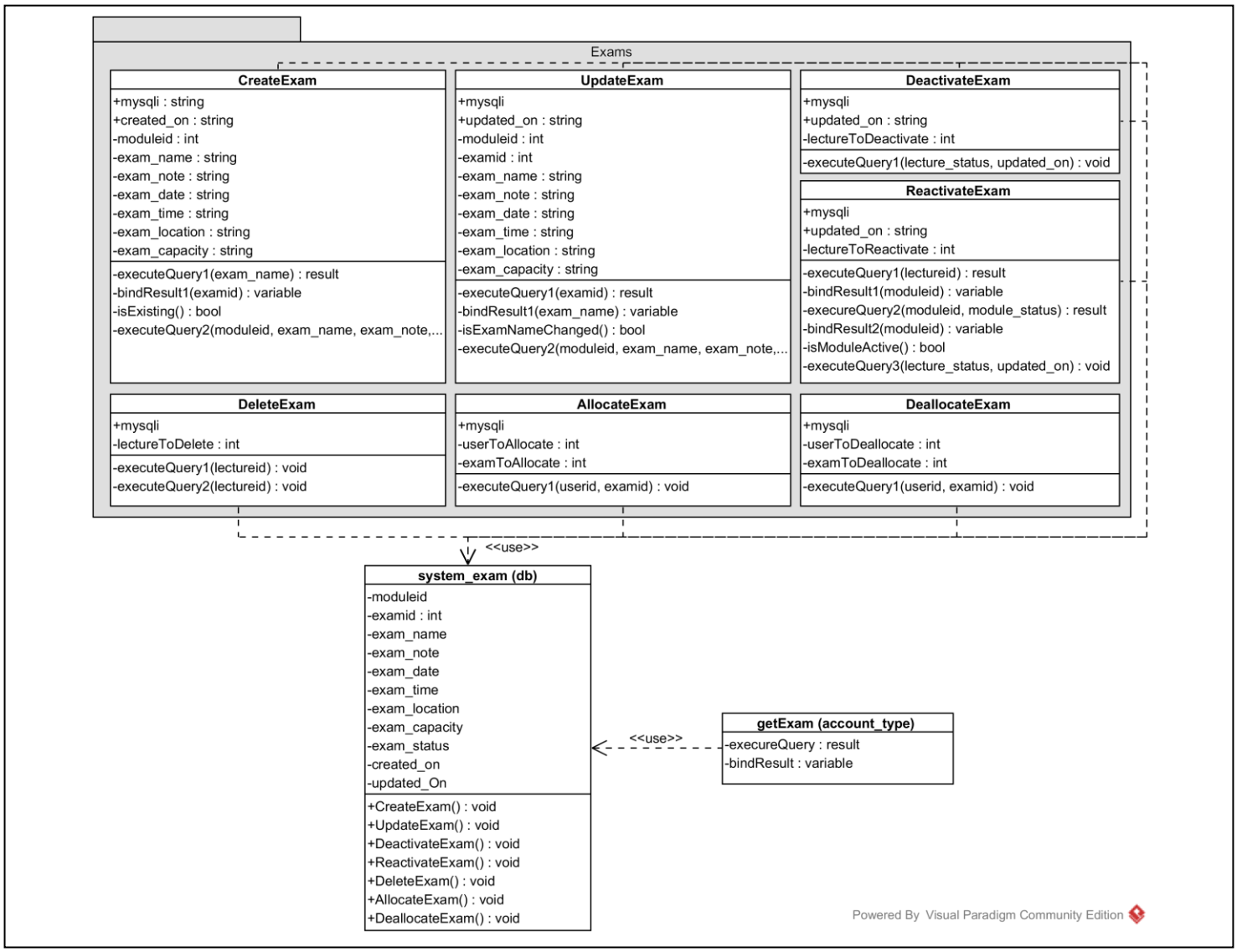


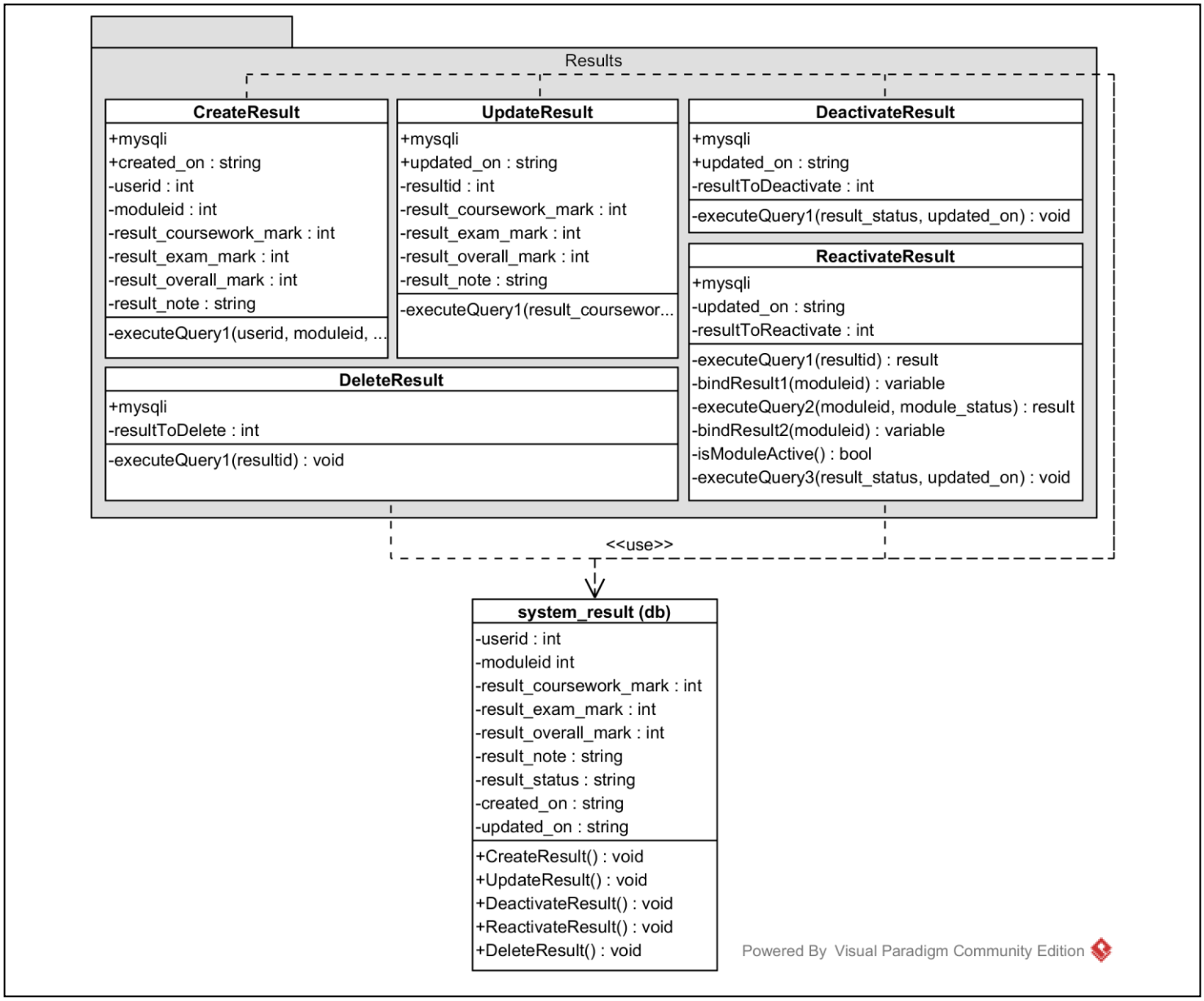


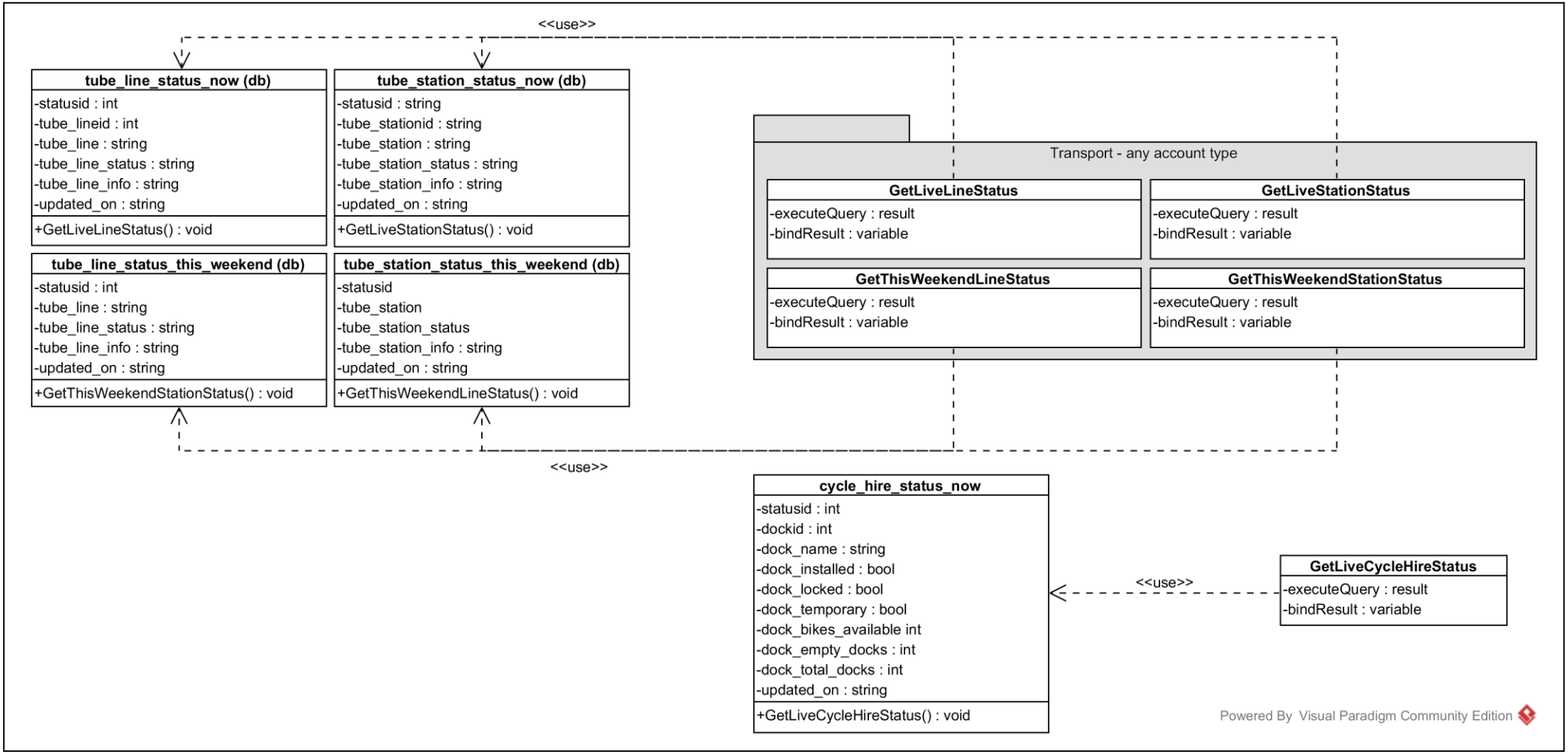

user_detail (db)
-userid : int
-firstname : string
-surname : string
-gender : string
-nationality : string
-studentno : int
-degree : string
-dateofbirth : string
-phonenumber : string
-address1 : string
-address2 : string
-town : string
-city : string
-country : string
-postcode : string
-user_status : string
-created_on : string
-updated_on : string
+SendPasswordToken() : void
user_token (db)
-userid : int
-token : string
-created_on : string
+SendPasswordToken() : void
Forgotten your password - any account type
SendPasswordToken
+mysqli
+created_on : string
-email : string
-isEmailValid() : bool
-executeQuery1(userid) : result
-bindResult1(userid) : variable
-isExisting() : bool
-executeQuery2(token, created_on) : void
-executeQuery3(firstname, surname) : result
-bindResult1(firstname, surname) : variable
-mail(email, subject, message, headers) : void
<<use>>
<<use>>
<<use>>
user_signin (db)
-userid : int
-account_type : string
-email : string
-password : string
-isSignedIn : tinyint
-created_on : string
-updated_on : string
+SendPasswordToken() : void
Powered By Visual Paradigm Community Edition

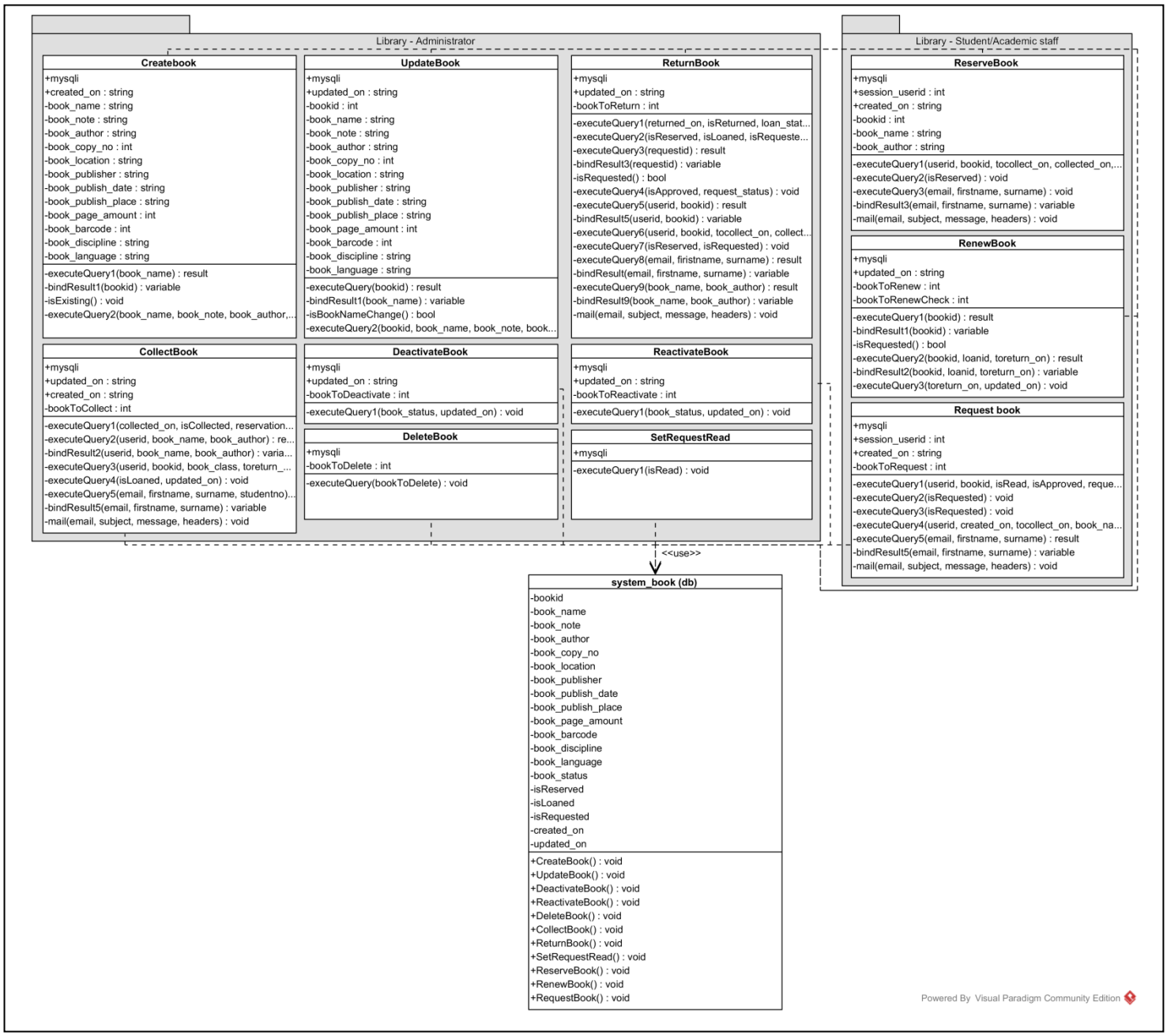

Password Reset - any account type
user_signin (db)
-userid : int
-email : string
-password : char
-isSignedIn : tinyint
-created_on : string
-updated_on : string
+ResetPassword() : void
ResetPassword
+mysqli
+updated_on : string
-token : string
-email : string
-password : char
-isEmailValid() : bool
-executeQuery1(userid) : result
-bindResult1(userid) : variable
-isExisting() : bool
-executeQuery2(token, firstname) : result
-bindResult2(token, firstname) : variable
-isTokenValid() : bool
-executeQuery3(password, updated_on) : void
-executeQuery4(token, created_on) : void
-mail(email, subject, message, headers) : void
user_token (db)
-userid : int
-token : char
-created_on : string
+ResetPassword() : void
<<use>>
<<use>>
Powered By Visual Paradigm Community Edition

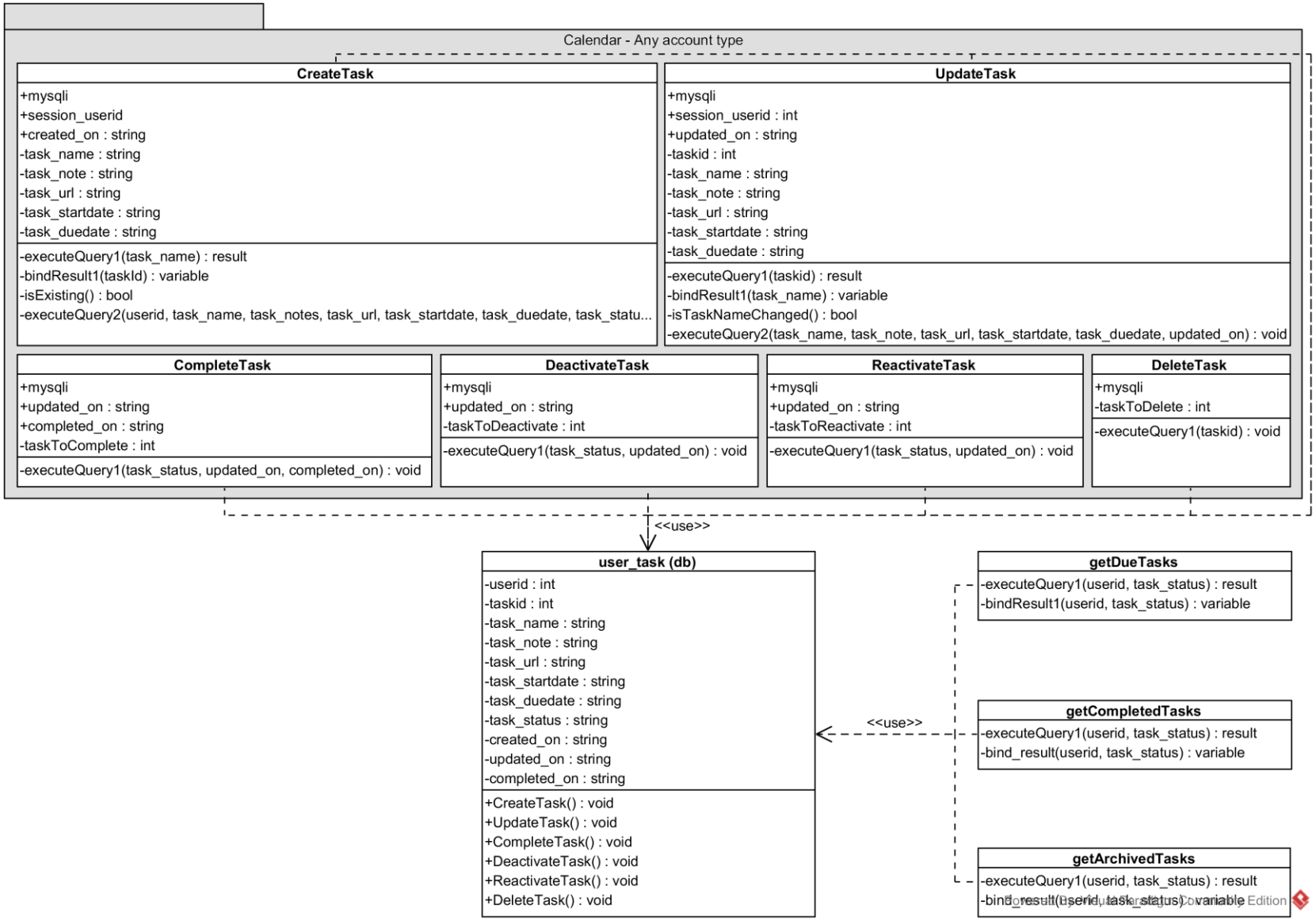

Modules
CreateModule
+mysqli
+created_on : string
-module_name : string
-module_note : string
-module_url : string
-executeQuery1(module_name) : result
-bindResult1(moduleid) : variable
-isExisting() : bool
-executeQuery2(module_name, module_notes...
DeactivateModule
+mysqli
+update_on : string
-moduleToDeactivate : int
-executeQuery1(module_status, updated_on) : void
-executeQuery2(lecture_status, updated_on) : void
-executeQuery3(tutorial_status, updated_on) : void
-executeQuery4(exam_status, updated_on) : void
-executeQuery5(result_status, updated_on) : void
DeleteModule
+mysqli
-moduleToDelete : int
-executeQuery1(moduleid) : void
-executeQuery2(moduleid) : void
-executeQuery3(moduleid) : void
-executeQuery4(moduleid) : void
-executeQuery5(moduleid) : result
-bindResult5(examid) : variable
-executeQuery6(examid) : void
-executeQuery7(moduleid) : void
-executeQuery8(moduleid) : result
-bindResult8(tutorialid) : variable
-executeQuery9(tutorialid) : void
-executeQuery10(moduleid) : void
-executeQuery11(moduleid) : result
-bindResult11(lectureid) : variable
-executeQuery12(lectureid) : void
-executeQuery13(moduleid) : void
-executeQuery14(moduleid) : void
-executeQuery15(moduleid) : void
AllocateModule
+mysqli
-userToAllocate : int
-moduleToAllocate : int
-executeQuery1(userid, moduleid) : void
DeallocateModule
+mysqli
-userToDeallocate
-moduleToDeallocate
+executeQuery1(userid, moduleid) : void
UpdateModule
+mysqli
+updated_on : string
-moduleid : int
-module_name : string
-module_note : string
-module_url : string
-executeQuery1(moduleid) : result
-bindResult1(module_name) : variable
-isModuleNameChanged() : boolean
-executeQuery2(module_name, module_notes...
ReactivateModule
+mysqli
+updated_on : string
-moduleToReactivate : int
-executeQuery1(module_status, updated_on) : void
-executeQuery2(lecture_status, updated_on)
-executeQuery3(tutorial_status, updated_on) : void
-executeQuery4(exam_status, updated_on) : void
-executeQuery5(result_status, updated_on) : void
Lectures
CreateLecture
+mysqli
+created_on : string
-moduleid : int
-lecture_name : string
-lecture_lecturer : int
-lecture_note : string
-lecture_day : string
-lecture_from_time : string
-lecture_to_time : string
-lecture_from_date : string
-lecture_to_date : string
-lecture_location : string
-lecture_capacity : string
-executeQuery1(lecture_name) : result
-bindResult1(lectureid) : variable
-isExisting() : boolean
-executeQuery2(moduleid, lecture_name, lectu...
UpdateLecture
+mysqli
+updated_on : string
-moduleid : int
-lectureid : int
-lecture_name : string
-lecture_lecturer : int
-lecture_note : string
-lecture_day : string
-lecture_from_time : string
-lecture_to_time : string
-lecture_from_date : string
-lecture_to_date : string
-lecture_location : string
-lecture_capacity : string
-executeQuery1(lectureid) : result
-bindResult1(lecture_name) : variable
-isLectureNameChanged() : boolean
-executeQuery2(moduleid, lecture_name, lecture_le...
DeactivateLecture
+mysqli
+updated_on : string
-lectureToDeactivate : int
-executeQuery1(lecture_status, updated_on) : void
ReactivateLecture
+mysqli
+updated_on : string
-lectureToReactivate : int
-executeQuery1(lectureid) : result
-bindResult1(moduleid) : variable
-executeQuery2(moduleid, module_status) : result
-bindResult2(moduleid) : variable
-isModuleActive() : bool
-executeQuery3(lecture_status, updated_on) : void
DeleteLecture
+mysqli
-lectureToDelete : int
-executeQuery1(lectureid) : void
-executeQuery2(lectureid) : void
AllocateLecture
+mysqli
-userToAllocate : int
-lectureToAllocate : int
-executeQuery(userid, lectureid) : void
DeallocateLecture
+mysqli
-userToDeallocate : int
-lectureToDeallocate : int
-executeQuery1(userid, lectureid) : void
Tutorials
CreateTutorial
+mysqli
+created_on : string
-moduleid : int
-tutorial_name : string
-tutorial_assistant : int
-tutorial_note : string
-tutorial_day : string
-tutorial_from_time : string
-tutorial_to_time : string
-tutorial_from_date : string
-tutorial_to_date : string
-tutorial_location : string
-tutorial_capacity : string
-executeQuery1(tutorial_name) : result
-bindResult1(tutorialid) : variable
-isExisting() : boolean
-executeQuery2(tutorial_name, tutorial_assista...
UpdateTutorial
+mysqli
+updated_on : string
-moduleid : int
-tutorialid : int
-tutorial_name : string
-tutorial_assistant : int
-tutorial_note : string
-tutorial_day : string
-tutorial_from_time : string
-tutorial_to_time : string
-tutorial_from_date : string
-tutorial_to_date : string
-tutorial_location : string
-tutorial_capacity : string
-execureQuery1(tutorialid) : void
-bindResult1(tutorial_name) : result
-isTutorialNameChanged() : bool
-execureQuery2(moduleid, tutorial_name, tutorial_a...
DeactivateTutorial
+mysqli
+updated_on : string
-tutorialToDeactivate : int
+execureQuery1(tutorial_status, updated_on) : void
ReactivateTutorial
+mysqli
+updated_on : string
-tutorialToReactivate : int
-execureQuery1(tutorialid) : result
-bindResult1(moduleid) : variable
-executeQuery2(moduleid, module_status) : result
-bindResult2(moduleid) : variable
-isModuleActive() : bool
-executeQuery3(tutorial_status, updated_on) : void
Delete tutorial
+mysqli
+tutorialToDelete : int
-execureQuery1(lectureid) : void
-execureQuery2(lectureid) : void
AllocateTutorial
+mysqli
-userToAllocate : int
-tutorialToDeallocate : int
-executeQuery1(userid, tutorialid) : void
DeallocateTutorial
+mysqli
-userToDeallocate : int
-tutorialToDeallocate : int
-executeQuery(userid, tutorialid) : void
<<use>>
<<use>>
system_module (db)
-moduleid : int
-module_name : string
-module_note : string
-module_url : string
-module_status : string
-created_on : string
-updated_on : string
+CreateModule() : void
+UpdateModule() : void
+DeactivateModule() : void
+ReactivateModule() : void
+DeleteModule() : void
+AllocateModule() : void
+DeallocateModule() : void
system_lecture (db)
-moduleid : int
-lectureid : int
-lecture_name : string
-lecture_lecturer : int
-lecture_note : string
-lecture_day : string
-lecture_from_time : string
-lecture_to_time : string
-lecture_from_date : string
-lecture_to_date : string
-lecture_location : string
-lecture_capacity : int
-lecture_status : string
-created_on : string
-updated_on : string
+CreateLecture() : void
+UpdateLecture() : void
+DeactivateLecture() : void
+ReactivateLecture() : void
+DeleteLecture() : void
+AllocateLecture() : void
+DeallocateLecture() : void
<<use>>
getLectures (day, account_type)
-executeQuery : result
-bindResult : variable
system_tutorial (db)
-moduleid : int
-tutorialid : int
-tutorial_name : string
-tutorial_assistant : int
-tutorial_note : string
-tutorial_day : string
-tutorial_from_time : string
-tutorial_to_time : string
-tutorial_from_date : string
-tutorial_to_date : string
-tutorial_location : string
-tutorial_capacity : int
-tutorial_status : string
-created_on : string
-updated_on : string
+CreateTutorial() : void
+UpdateTutorial() : void
+DeactivateTutorial() : void
+ReactivateTutorial() : void
+DeleteTutorial() : void
+AllocateTutorial() : void
+DeallocateTutorial() : void
<<use>>
getTutorials (day, account_type)
-execureQuery : result
-bindResult : variable
<<use>>
Powered By Visual Paradigm Community Edition

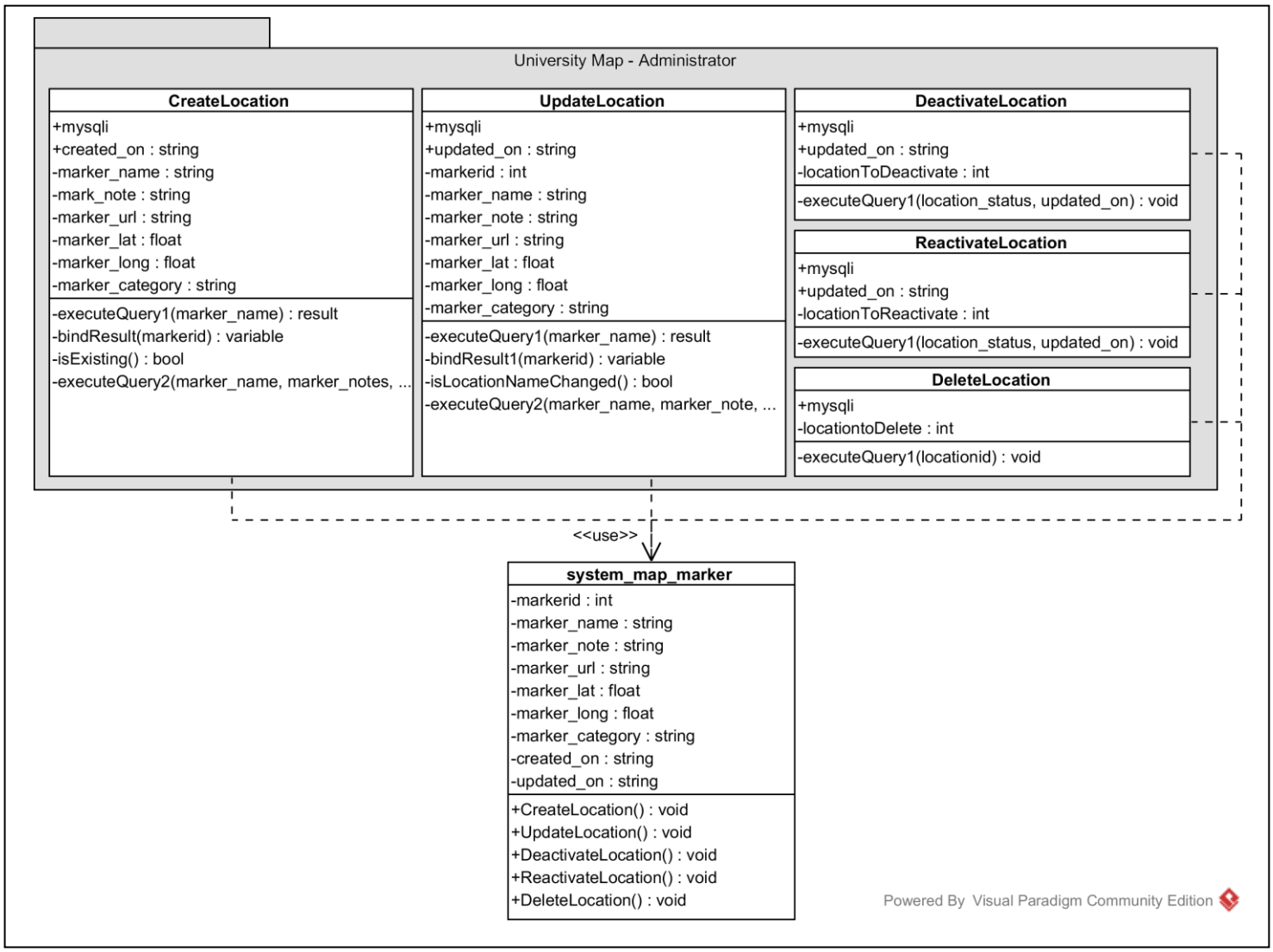

Exams
CreateExam
+mysqli : string
+created_on : string
-moduleid : int
-exam_name : string
-exam_note : string
-exam_date : string
-exam_time : string
-exam_location : string
-exam_capacity : string
-executeQuery1(exam_name) : result
-bindResult1(examid) : variable
-isExisting() : bool
-executeQuery2(moduleid, exam_name, exam_note,...
UpdateExam
+mysqli
+updated_on : string
-moduleid : int
-examid : int
-exam_name : string
-exam_note : string
-exam_date : string
-exam_time : string
-exam_location : string
-exam_capacity : string
-executeQuery1(examid) : result
-bindResult1(exam_name) : variable
-isExamNameChanged() : bool
-executeQuery2(moduleid, exam_name, exam_note,...
DeactivateExam
+mysqli
+updated_on : string
-lectureToDeactivate : int
-executeQuery1(lecture_status, updated_on) : void
ReactivateExam
+mysqli
+updated_on : string
-lectureToReactivate : int
-executeQuery1(lectureid) : result
-bindResult1(moduleid) : variable
-execureQuery2(moduleid, module_status) : result
-bindResult2(moduleid) : variable
-isModuleActive() : bool
-executeQuery3(lecture_status, updated_on) : void
DeleteExam
+mysqli
-lectureToDelete : int
-executeQuery1(lectureid) : void
-executeQuery2(lectureid) : void
AllocateExam
+mysqli
-userToAllocate : int
-examToAllocate : int
-executeQuery1(userid, examid) : void
DeallocateExam
+mysqli
-userToDeallocate : int
-examToDeallocate : int
-executeQuery1(userid, examid) : void
<<use>>
system_exam (db)
-moduleid
-examid : int
-exam_name
-exam_note
-exam_date
-exam_time
-exam_location
-exam_capacity
-exam_status
-created_on
-updated_On
+CreateExam() : void
+UpdateExam() : void
+DeactivateExam() : void
+ReactivateExam() : void
+DeleteExam() : void
+AllocateExam() : void
+DeallocateExam() : void
<<use>>
getExam (account_type)
-execureQuery : result
-bindResult : variable
Powered By Visual Paradigm Community Edition

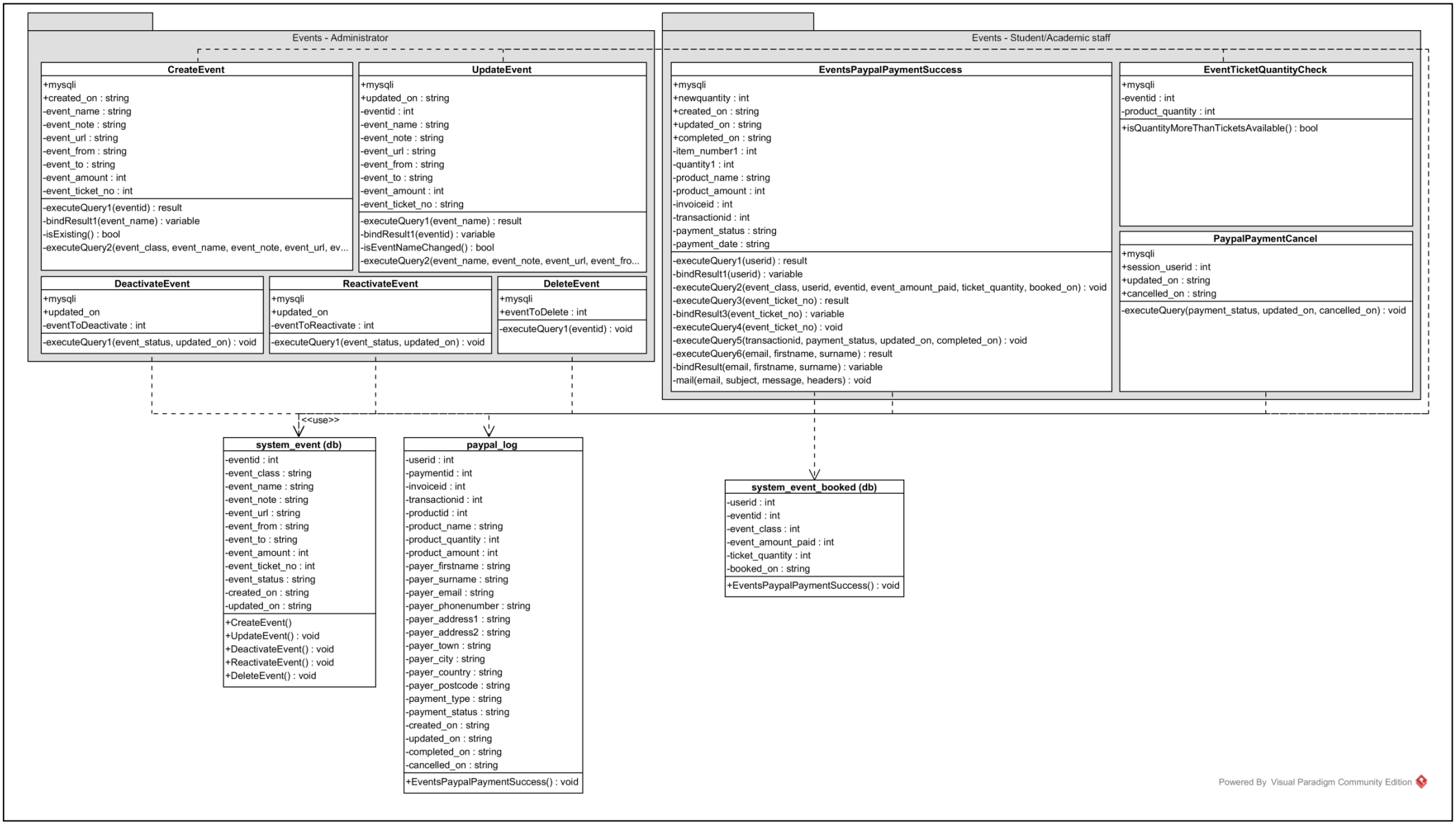

Results
CreateResult
+mysqli
+created_on : string
-userid : int
-moduleid : int
-result_coursework_mark : int
-result_exam_mark : int
-result_overall_mark : int
-result_note : string
-executeQuery1(userid, moduleid, ...
UpdateResult
+mysqli
+updated_on : string
-resultid : int
-result_coursework_mark : int
-result_exam_mark : int
-result_overall_mark : int
-result_note : string
-executeQuery1(result_coursewor...
DeactivateResult
+mysqli
+updated_on : string
-resultToDeactivate : int
-executeQuery1(result_status, updated_on) : void
ReactivateResult
+mysqli
-updated_on : string
-resultToReactivate : int
-executeQuery1(resultid) : result
-bindResult1(moduleid) : variable
-executeQuery2(moduleid, module_status) : result
-bindResult2(moduleid) : variable
-isModuleActive() : bool
-executeQuery3(result_status, updated_on) : void
DeleteResult
+mysqli
-resultToDelete : int
-executeQuery1(resultid) : void
<<use>>
system_result (db)
-userid : int
-moduleid int
-result_coursework_mark : int
-result_exam_mark : int
-result_overall_mark : int
-result_note : string
-result_status : string
-created_on : string
-updated_on : string
+CreateResult() : void
+UpdateResult() : void
+DeactivateResult() : void
+ReactivateResult() : void
+DeleteResult() : void
Powered By Visual Paradigm Community Edition

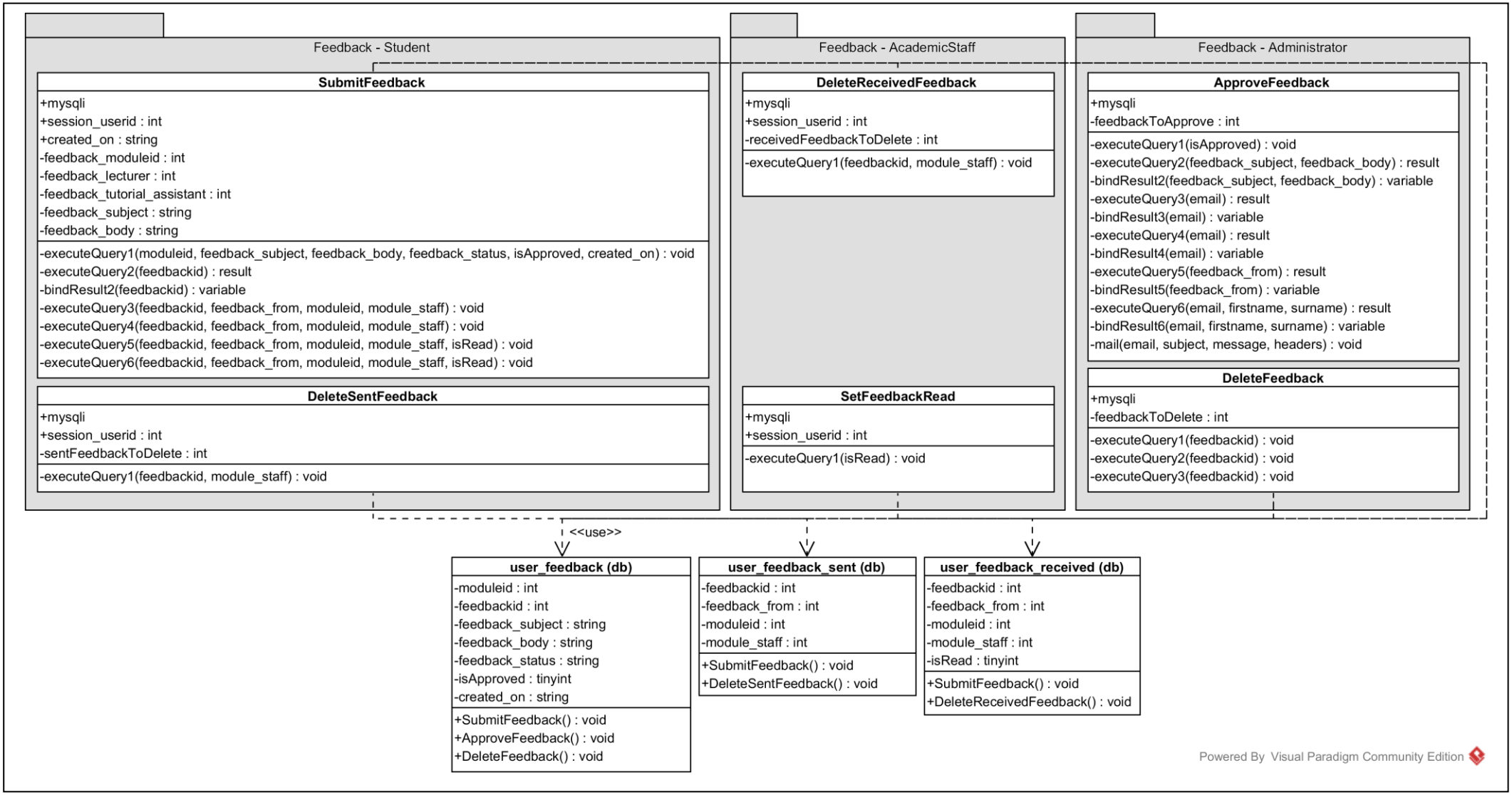

<<use>>
tube_line_status_now (db)
-statusid : int
-tube_lineid : int
-tube_line : string
-tube_line_status : string
-tube_line_info : string
-updated_on : string
+GetLiveLineStatus() : void
tube_station_status_now (db)
-statusid : string
-tube_stationid : string
-tube_station : string
-tube_station_status : string
-tube_station_info : string
-updated_on : string
+GetLiveStationStatus() : void
tube_line_status_this_weekend (db)
-statusid : int
-tube_line : string
-tube_line_status : string
-tube_line_info : string
-updated_on : string
+GetThisWeekendStationStatus() : void
tube_station_status_this_weekend (db)
-statusid
-tube_station
-tube_station_status
-tube_station_info : string
-updated_on : string
+GetThisWeekendLineStatus() : void
Transport - any account type
GetLiveLineStatus
-executeQuery : result
-bindResult : variable
GetLiveStationStatus
-executeQuery : result
-bindResult : variable
GetThisWeekendLineStatus
-executeQuery : result
-bindResult : variable
GetThisWeekendStationStatus
-executeQuery : result
-bindResult : variable
<<use>>
cycle_hire_status_now
-statusid : int
-dockid : int
-dock_name : string
-dock_installed : bool
-dock_locked : bool
-dock_temporary : bool
-dock_bikes_available int
-dock_empty_docks : int
-dock_total_docks : int
-updated_on : string
+GetLiveCycleHireStatus() : void
<<use>>
GetLiveCycleHireStatus
-executeQuery : result
-bindResult : variable
Powered By Visual Paradigm Community Edition

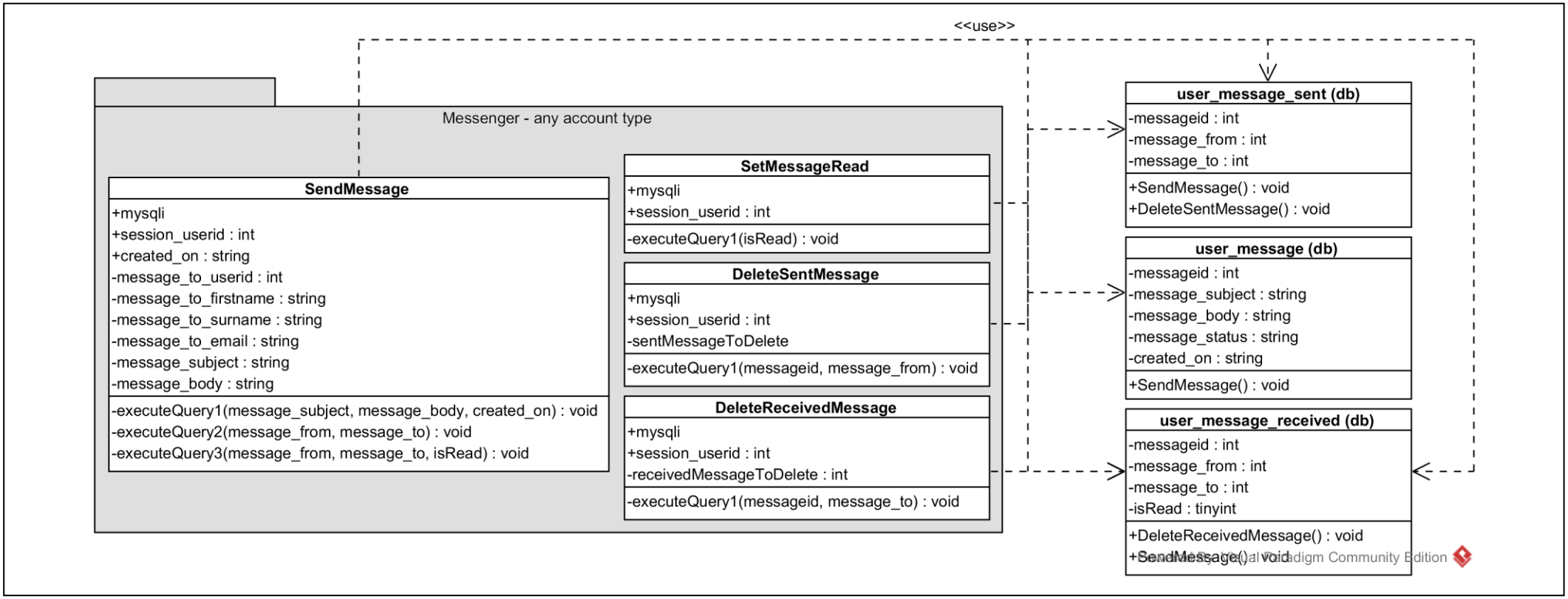

Library - Administrator
Createbook
+mysqli
+created_on : string
-book_name : string
-book_note : string
-book_author : string
-book_copy_no : int
-book_location : string
-book_publisher : string
-book_publish_date : string
-book_publish_place : string
-book_page_amount : int
-book_barcode : int
-book_discipline : string
-book_language : string
-executeQuery1(book_name) : result
-bindResult1(bookid) : variable
-isExisting() : void
-executeQuery2(book_name, book_note, book_author,...
UpdateBook
+mysqli
+updated_on : string
-bookid : int
-book_name : string
-book_note : string
-book_author : string
-book_copy_no : int
-book_location : string
-book_publisher : string
-book_publish_date : string
-book_publish_place : string
-book_page_amount : int
-book_barcode : int
-book_discipline : string
-book_language : string
-executeQuery(bookid) : result
-bindResult1(book_name) : variable
-isBookNameChange() : bool
-executeQuery2(bookid, book_name, book_note, book...
ReturnBook
+mysqli
+updated_on : string
-bookToReturn : int
-executeQuery1(returned_on, isReturned, loan_stat...
-executeQuery2(isReserved, isLoaned, isRequeste...
-executeQuery3(requestid) : result
-bindResult3(requestid) : variable
-isRequested() : bool
-executeQuery4(isApproved, request_status) : void
-executeQuery5(userid, bookid) : result
-bindResult5(userid, bookid) : variable
-executeQuery6(userid, bookid, tocollect_on, collect...
-executeQuery7(isReserved, isRequested) : void
-executeQuery8(email, firstname, surname) : result
-bindResult(email, firstname, surname) : variable
-executeQuery9(book_name, book_author) : result
-bindResult9(book_name, book_author) : variable
-mail(email, subject, message, headers) : void
CollectBook
+mysqli
+updated_on : string
+created_on : string
-bookToCollect : int
-executeQuery1(collected_on, isCollected, reservation...
-executeQuery2(userid, book_name, book_author) : re...
-bindResult2(userid, book_name, book_author) : varia...
-executeQuery3(userid, bookid, book_class, toreturn_...
-executeQuery4(isLoaned, updated_on) : void
-executeQuery5(email, firstname, surname, studentno)...
-bindResult5(email, firstname, surname) : variable
-mail(email, subject, message, headers) : void
DeactivateBook
+mysqli
+updated_on : string
-bookToDeactivate : int
-executeQuery1(book_status, updated_on) : void
ReactivateBook
+mysqli
+updated_on : string
-bookToReactivate : int
-executeQuery1(book_status, updated_on) : void
DeleteBook
+mysqli
-bookToDelete : int
-executeQuery(bookToDelete) : void
SetRequestRead
+mysqli
-executeQuery1(isRead) : void
Library - Student/Academic staff
ReserveBook
+mysqli
+session_userid : int
+created_on : string
-bookid : int
-book_name : string
-book_author : string
-executeQuery1(userid, bookid, tocollect_on, collected_on,...
-executeQuery2(isReserved) : void
-executeQuery3(email, firstname, surname) : void
-bindResult3(email, firstname, surname) : variable
-mail(email, subject, message, headers) : void
RenewBook
+mysqli
+updated_on : string
-bookToRenew : int
-bookToRenewCheck : int
-executeQuery1(bookid) : result
-bindResult1(bookid) : variable
-isRequested() : bool
-executeQuery2(bookid, loanid, toreturn_on) : result
-bindResult2(bookid, loanid, toreturn_on) : variable
-executeQuery3(toreturn_on, updated_on) : void
Request book
+mysqli
+session_userid : int
+created_on : string
-bookToRequest : int
-executeQuery1(userid, bookid, isRead, isApproved, reque...
-executeQuery2(isRequested) : void
-executeQuery3(isRequested) : void
-executeQuery4(userid, created_on, tocollect_on, book_na...
-executeQuery5(email, firstname, surname) : result
-bindResult5(email, firstname, surname) : variable
-mail(email, subject, message, headers) : void
<<use>>
system_book (db)
-bookid
-book_name
-book_note
-book_author
-book_copy_no
-book_location
-book_publisher
-book_publish_date
-book_publish_place
-book_page_amount
-book_barcode
-book_discipline
-book_language
-book_status
-isReserved
-isLoaned
-isRequested
-created_on
-updated_on
+CreateBook() : void
+UpdateBook() : void
+DeactivateBook() : void
+ReactivateBook() : void
+DeleteBook() : void
+CollectBook() : void
+ReturnBook() : void
+SetRequestRead() : void
+ReserveBook() : void
+RenewBook() : void
+RequestBook() : void
Powered By Visual Paradigm Community Edition

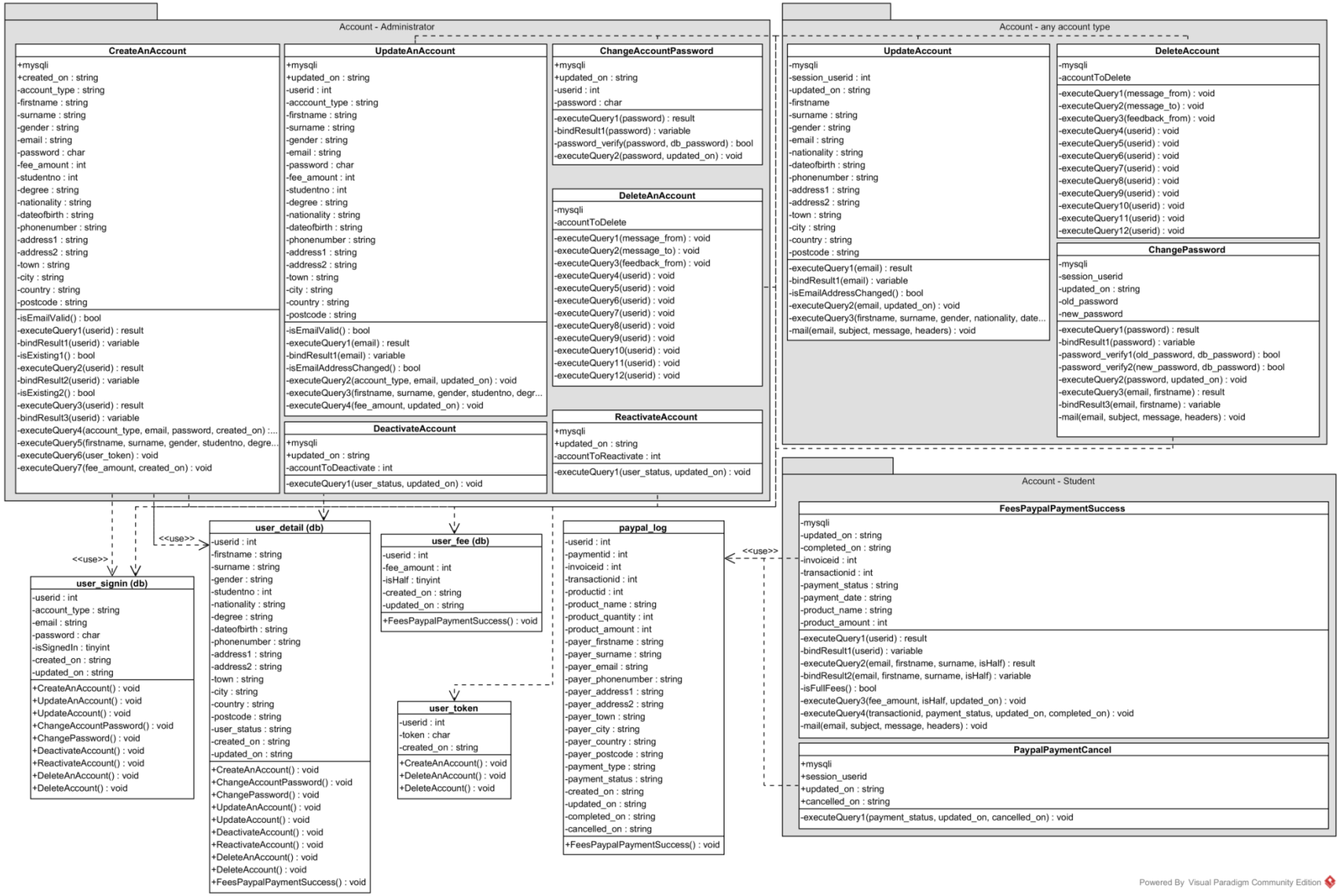

Calendar - Any account type
CreateTask
+mysqli
+session_userid
+created_on : string
-task_name : string
-task_note : string
-task_url : string
-task_startdate : string
-task_duedate : string
-executeQuery1(task_name) : result
-bindResult1(taskId) : variable
-isExisting() : bool
-executeQuery2(userid, task_name, task_notes, task_url, task_startdate, task_duedate, task_statu...
UpdateTask
+mysqli
+session_userid : int
+updated_on : string
-taskid : int
-task_name : string
-task_note : string
-task_url : string
-task_startdate : string
-task_duedate : string
-executeQuery1(taskid) : result
-bindResult1(task_name) : variable
-isTaskNameChanged() : bool
-executeQuery2(task_name, task_note, task_url, task_startdate, task_duedate, updated_on) : void
CompleteTask
+mysqli
+updated_on : string
+completed_on : string
-taskToComplete : int
-executeQuery1(task_status, updated_on, completed_on) : void
DeactivateTask
+mysqli
+updated_on : string
-taskToDeactivate : int
-executeQuery1(task_status, updated_on) : void
ReactivateTask
+mysqli
+updated_on : string
-taskToReactivate : int
-executeQuery1(task_status, updated_on) : void
DeleteTask
+mysqli
-taskToDelete : int
-executeQuery1(taskid) : void
<<use>>
user_task (db)
-userid : int
-taskid : int
-task_name : string
-task_note : string
-task_url : string
-task_startdate : string
-task_duedate : string
-task_status : string
-created_on : string
-updated_on : string
-completed_on : string
+CreateTask() : void
+UpdateTask() : void
+CompleteTask() : void
+DeactivateTask() : void
+ReactivateTask() : void
+DeleteTask() : void
<<use>>
getDueTasks
-executeQuery1(userid, task_status) : result
-bindResult1(userid, task_status) : variable
getCompletedTasks
-executeQuery1(userid, task_status) : result
-bind_result(userid, task_status) : variable
getArchivedTasks
-executeQuery1(userid, task_status) : result
-bind_result(userid, task_status) : variable


University Map - Administrator
CreateLocation
+mysqli
+created_on : string
-marker_name : string
-mark_note : string
-marker_url : string
-marker_lat : float
-marker_long : float
-marker_category : string
-executeQuery1(marker_name) : result
-bindResult(markerid) : variable
-isExisting() : bool
-executeQuery2(marker_name, marker_notes, ...
UpdateLocation
+mysqli
+updated_on : string
-markerid : int
-marker_name : string
-marker_note : string
-marker_url : string
-marker_lat : float
-marker_long : float
-marker_category : string
-executeQuery1(marker_name) : result
-bindResult1(markerid) : variable
-isLocationNameChanged() : bool
-executeQuery2(marker_name, marker_note, ...
DeactivateLocation
+mysqli
+updated_on : string
-locationToDeactivate : int
-executeQuery1(location_status, updated_on) : void
ReactivateLocation
+mysqli
+updated_on : string
-locationToReactivate : int
-executeQuery1(location_status, updated_on) : void
DeleteLocation
+mysqli
-locationtoDelete : int
-executeQuery1(locationid) : void
<<use>>
system_map_marker
-markerid : int
-marker_name : string
-marker_note : string
-marker_url : string
-marker_lat : float
-marker_long : float
-marker_category : string
-created_on : string
-updated_on : string
+CreateLocation() : void
+UpdateLocation() : void
+DeactivateLocation() : void
+ReactivateLocation() : void
+DeleteLocation() : void
Powered By Visual Paradigm Community Edition


Events - Administrator
Events - Student/Academic staff
CreateEvent
+mysqli
+created_on : string
-event_name : string
-event_note : string
-event_url : string
-event_from : string
-event_to : string
-event_amount : int
-event_ticket_no : int
-executeQuery1(eventid) : result
-bindResult1(event_name) : variable
-isExisting() : bool
-executeQuery2(event_class, event_name, event_note, event_url, ev...
UpdateEvent
+mysqli
+updated_on : string
-eventid : int
-event_name : string
-event_note : string
-event_url : string
-event_from : string
-event_to : string
-event_amount : int
-event_ticket_no : string
-executeQuery1(event_name) : result
-bindResult1(eventid) : variable
-isEventNameChanged() : bool
-executeQuery2(event_name, event_note, event_url, event_fro...
DeactivateEvent
+mysqli
+updated_on
-eventToDeactivate : int
-executeQuery1(event_status, updated_on) : void
ReactivateEvent
+mysqli
+updated_on
-eventToReactivate : int
-executeQuery1(event_status, updated_on) : void
DeleteEvent
+mysqli
+eventToDelete : int
-executeQuery1(eventid) : void
EventsPaypalPaymentSuccess
+mysqli
+newquantity : int
+created_on : string
+updated_on : string
+completed_on : string
-item_number1 : int
-quantity1 : int
-product_name : string
-product_amount : int
-invoiceid : int
-transactionid : int
-payment_status : string
-payment_date : string
-executeQuery1(userid) : result
-bindResult1(userid) : variable
-executeQuery2(event_class, userid, eventid, event_amount_paid, ticket_quantity, booked_on) : void
-executeQuery3(event_ticket_no) : result
-bindResult3(event_ticket_no) : variable
-executeQuery4(event_ticket_no) : void
-executeQuery5(transactionid, payment_status, updated_on, completed_on) : void
-executeQuery6(email, firstname, surname) : result
-bindResult(email, firstname, surname) : variable
-mail(email, subject, message, headers) : void
EventTicketQuantityCheck
+mysqli
-eventid : int
-product_quantity : int
+isQuantityMoreThanTicketsAvailable() : bool
PaypalPaymentCancel
+mysqli
+session_userid : int
+updated_on : string
+cancelled_on : string
-executeQuery(payment_status, updated_on, cancelled_on) : void
<<use>>
system_event (db)
-eventid : int
-event_class : string
-event_name : string
-event_note : string
-event_url : string
-event_from : string
-event_to : string
-event_amount : int
-event_ticket_no : int
-event_status : string
-created_on : string
-updated_on : string
+CreateEvent()
+UpdateEvent() : void
+DeactivateEvent() : void
+ReactivateEvent() : void
+DeleteEvent() : void
paypal_log
-userid : int
-paymentid : int
-invoiceid : int
-transactionid : int
-productid : int
-product_name : string
-product_quantity : int
-product_amount : int
-payer_firstname : string
-payer_surname : string
-payer_email : string
-payer_phonenumber : string
-payer_address1 : string
-payer_address2 : string
-payer_town : string
-payer_city : string
-payer_country : string
-payer_postcode : string
-payment_type : string
-payment_status : string
-created_on : string
-updated_on : string
-completed_on : string
-cancelled_on : string
+EventsPaypalPaymentSuccess() : void
system_event_booked (db)
-userid : int
-eventid : int
-event_class : int
-event_amount_paid : int
-ticket_quantity : int
-booked_on : string
+EventsPaypalPaymentSuccess() : void
Powered By Visual Paradigm Community Edition


Feedback - Student
SubmitFeedback
+mysqli
+session_userid : int
+created_on : string
-feedback_moduleid : int
-feedback_lecturer : int
-feedback_tutorial_assistant : int
-feedback_subject : string
-feedback_body : string
-executeQuery1(moduleid, feedback_subject, feedback_body, feedback_status, isApproved, created_on) : void
-executeQuery2(feedbackid) : result
-bindResult2(feedbackid) : variable
-executeQuery3(feedbackid, feedback_from, moduleid, module_staff) : void
-executeQuery4(feedbackid, feedback_from, moduleid, module_staff) : void
-executeQuery5(feedbackid, feedback_from, moduleid, module_staff, isRead) : void
-executeQuery6(feedbackid, feedback_from, moduleid, module_staff, isRead) : void
DeleteSentFeedback
+mysqli
+session_userid : int
-sentFeedbackToDelete : int
-executeQuery1(feedbackid, module_staff) : void
Feedback - AcademicStaff
DeleteReceivedFeedback
+mysqli
+session_userid : int
-receivedFeedbackToDelete : int
-executeQuery1(feedbackid, module_staff) : void
SetFeedbackRead
+mysqli
+session_userid : int
-executeQuery1(isRead) : void
Feedback - Administrator
ApproveFeedback
+mysqli
-feedbackToApprove : int
-executeQuery1(isApproved) : void
-executeQuery2(feedback_subject, feedback_body) : result
-bindResult2(feedback_subject, feedback_body) : variable
-executeQuery3(email) : result
-bindResult3(email) : variable
-executeQuery4(email) : result
-bindResult4(email) : variable
-executeQuery5(feedback_from) : result
-bindResult5(feedback_from) : variable
-executeQuery6(email, firstname, surname) : result
-bindResult6(email, firstname, surname) : variable
-mail(email, subject, message, headers) : void
DeleteFeedback
+mysqli
-feedbackToDelete : int
-executeQuery1(feedbackid) : void
-executeQuery2(feedbackid) : void
-executeQuery3(feedbackid) : void
<<use>>
user_feedback (db)
-moduleid : int
-feedbackid : int
-feedback_subject : string
-feedback_body : string
-feedback_status : string
-isApproved : tinyint
-created_on : string
+SubmitFeedback() : void
+ApproveFeedback() : void
+DeleteFeedback() : void
user_feedback_sent (db)
-feedbackid : int
-feedback_from : int
-moduleid : int
-module_staff : int
+SubmitFeedback() : void
+DeleteSentFeedback() : void
user_feedback_received (db)
-feedbackid : int
-feedback_from : int
-moduleid : int
-module_staff : int
-isRead : tinyint
+SubmitFeedback() : void
+DeleteReceivedFeedback() : void
Powered By Visual Paradigm Community Edition


<<use>>
Messenger - any account type
SendMessage
+mysqli
+session_userid : int
+created_on : string
-message_to_userid : int
-message_to_firstname : string
-message_to_surname : string
-message_to_email : string
-message_subject : string
-message_body : string
-executeQuery1(message_subject, message_body, created_on) : void
-executeQuery2(message_from, message_to) : void
-executeQuery3(message_from, message_to, isRead) : void
SetMessageRead
+mysqli
+session_userid : int
-executeQuery1(isRead) : void
DeleteSentMessage
+mysqli
+session_userid : int
-sentMessageToDelete
-executeQuery1(messageid, message_from) : void
DeleteReceivedMessage
+mysqli
+session_userid : int
-receivedMessageToDelete : int
-executeQuery1(messageid, message_to) : void
user_message_sent (db)
-messageid : int
-message_from : int
-message_to : int
+SendMessage() : void
+DeleteSentMessage() : void
user_message (db)
-messageid : int
-message_subject : string
-message_body : string
-message_status : string
-created_on : string
+SendMessage() : void
user_message_received (db)
-messageid : int
-message_from : int
-message_to : int
-isRead : tinyint
+DeleteReceivedMessage() : void
+SendMessage() : void


Account - Administrator
CreateAnAccount
+mysqli
+created_on : string
-account_type : string
-firstname : string
-surname : string
-gender : string
-email : string
-password : char
-fee_amount : int
-studentno : int
-degree : string
-nationality : string
-dateofbirth : string
-phonenumber : string
-address1 : string
-address2 : string
-town : string
-city : string
-country : string
-postcode : string
-isEmailValid() : bool
-executeQuery1(userid) : result
-bindResult1(userid) : variable
-isExisting1() : bool
-executeQuery2(userid) : result
-bindResult2(userid) : variable
-isExisting2() : bool
-executeQuery3(userid) : result
-bindResult3(userid) : variable
-executeQuery4(account_type, email, password, created_on) :...
-executeQuery5(firstname, surname, gender, studentno, degre...
-executeQuery6(user_token) : void
-executeQuery7(fee_amount, created_on) : void
UpdateAnAccount
+mysqli
+updated_on : string
-userid : int
-acccount_type : string
-firstname : string
-surname : string
-gender : string
-email : string
-password : char
-fee_amount : int
-studentno : int
-degree : string
-nationality : string
-dateofbirth : string
-phonenumber : string
-address1 : string
-address2 : string
-town : string
-city : string
-country : string
-postcode : string
-isEmailValid() : bool
-executeQuery1(email) : result
-bindResult1(email) : variable
-isEmailAddressChanged() : bool
-executeQuery2(account_type, email, updated_on) : void
-executeQuery3(firstname, surname, gender, studentno, degr...
-executeQuery4(fee_amount, updated_on) : void
ChangeAccountPassword
+mysqli
+updated_on : string
-userid : int
-password : char
-executeQuery1(password) : result
-bindResult1(password) : variable
-password_verify(password, db_password) : bool
-executeQuery2(password, updated_on) : void
DeleteAnAccount
-mysqli
-accountToDelete
-executeQuery1(message_from) : void
-executeQuery2(message_to) : void
-executeQuery3(feedback_from) : void
-executeQuery4(userid) : void
-executeQuery5(userid) : void
-executeQuery6(userid) : void
-executeQuery7(userid) : void
-executeQuery8(userid) : void
-executeQuery9(userid) : void
-executeQuery10(userid) : void
-executeQuery11(userid) : void
-executeQuery12(userid) : void
DeactivateAccount
+mysqli
+updated_on : string
-accountToDeactivate : int
-executeQuery1(user_status, updated_on) : void
ReactivateAccount
+mysqli
+updated_on : string
-accountToReactivate : int
-executeQuery1(user_status, updated_on) : void
Account - any account type
UpdateAccount
-mysqli
-session_userid : int
-updated_on : string
-firstname
-surname : string
-gender : string
-email : string
-nationality : string
-dateofbirth : string
-phonenumber : string
-address1 : string
-address2 : string
-town : string
-city : string
-country : string
-postcode : string
-executeQuery1(email) : result
-bindResult1(email) : variable
-isEmailAddressChanged() : bool
-executeQuery2(email, updated_on) : void
-executeQuery3(firstname, surname, gender, nationality, date...
-mail(email, subject, message, headers) : void
DeleteAccount
-mysqli
-accountToDelete
-executeQuery1(message_from) : void
-executeQuery2(message_to) : void
-executeQuery3(feedback_from) : void
-executeQuery4(userid) : void
-executeQuery5(userid) : void
-executeQuery6(userid) : void
-executeQuery7(userid) : void
-executeQuery8(userid) : void
-executeQuery9(userid) : void
-executeQuery10(userid) : void
-executeQuery11(userid) : void
-executeQuery12(userid) : void
ChangePassword
-mysqli
-session_userid
-updated_on : string
-old_password
-new_password
-executeQuery1(password) : result
-bindResult1(password) : variable
-password_verify1(old_password, db_password) : bool
-password_verify2(new_password, db_password) : bool
-executeQuery2(password, updated_on) : void
-executeQuery3(email, firstname) : result
-bindResult3(email, firstname) : variable
-mail(email, subject, message, headers) : void
Account - Student
FeesPaypalPaymentSuccess
-mysqli
-updated_on : string
-completed_on : string
-invoiceid : int
-transactionid : int
-payment_status : string
-payment_date : string
-product_name : string
-product_amount : int
-executeQuery1(userid) : result
-bindResult1(userid) : variable
-executeQuery2(email, firstname, surname, isHalf) : result
-bindResult2(email, firstname, surname, isHalf) : variable
-isFullFees() : bool
-executeQuery3(fee_amount, isHalf, updated_on) : void
-executeQuery4(transactionid, payment_status, updated_on, completed_on) : void
-mail(email, subject, message, headers) : void
PaypalPaymentCancel
+mysqli
+session_userid
+updated_on : string
+cancelled_on : string
-executeQuery1(payment_status, updated_on, cancelled_on) : void
<<use>>
<<use>>
<<use>>
user_signin (db)
-userid : int
-account_type : string
-email : string
-password : char
-isSignedIn : tinyint
-created_on : string
-updated_on : string
+CreateAnAccount() : void
+UpdateAnAccount() : void
+UpdateAccount() : void
+ChangeAccountPassword() : void
+ChangePassword() : void
+DeactivateAccount() : void
+ReactivateAccount() : void
+DeleteAnAccount() : void
+DeleteAccount() : void
user_detail (db)
-userid : int
-firstname : string
-surname : string
-gender : string
-studentno : int
-nationality : string
-degree : string
-dateofbirth : string
-phonenumber : string
-address1 : string
-address2 : string
-town : string
-city : string
-country : string
-postcode : string
-user_status : string
-created_on : string
-updated_on : string
+CreateAnAccount() : void
+ChangeAccountPassword() : void
+ChangePassword() : void
+UpdateAnAccount() : void
+UpdateAccount() : void
+DeactivateAccount() : void
+ReactivateAccount() : void
+DeleteAnAccount() : void
+DeleteAccount() : void
+FeesPaypalPaymentSuccess() : void
user_fee (db)
-userid : int
-fee_amount : int
-isHalf : tinyint
-created_on : string
-updated_on : string
+FeesPaypalPaymentSuccess() : void
user_token
-userid : int
-token : char
-created_on : string
+CreateAnAccount() : void
+DeleteAnAccount() : void
+DeleteAccount() : void
paypal_log
-userid : int
-paymentid : int
-invoiceid : int
-transactionid : int
-productid : int
-product_name : string
-product_quantity : int
-product_amount : int
-payer_firstname : string
-payer_surname : string
-payer_email : string
-payer_phonenumber : string
-payer_address1 : string
-payer_address2 : string
-payer_town : string
-payer_city : string
-payer_country : string
-payer_postcode : string
-payment_type : string
-payment_status : string
-created_on : string
-updated_on : string
-completed_on : string
-cancelled_on : string
+FeesPaypalPaymentSuccess() : void
Powered By Visual Paradigm Community Edition

4. CREATE and DROP database statements

```
#Feedback
DROP TABLE user_feedback_sent;
DROP TABLE user_feedback_received;
DROP TABLE user_feedback;

#Exams
DROP TABLE user_exam;
DROP TABLE system_exam;

#Results
DROP TABLE user_result;

#Timetable
DROP TABLE user_lecture;
DROP TABLE system_lecture;
DROP TABLE user_tutorial;
DROP TABLE system_tutorial;
DROP TABLE user_module;
DROP TABLE system_module;

#Transport
DROP TABLE cycle_hire_status_now;
DROP TABLE tube_station_status_this_weekend;
DROP TABLE tube_line_status_this_weekend;
DROP TABLE tube_station_status_now;
DROP TABLE tube_line_status_now;

#Library
DROP TABLE system_book_reserved;
DROP TABLE system_book_loaned;
DROP TABLE system_book_requested;
DROP TABLE system_book;

#Calendar
DROP TABLE user_task;

#University Map
DROP TABLE system_map_marker;

#Events
DROP TABLE system_event_booked;
DROP TABLE system_event;

#Messenger
DROP TABLE user_message_sent;
DROP TABLE user_message_received;
DROP TABLE user_message;

#Account
DROP TABLE paypal_log;
DROP TABLE user_fee;
DROP TABLE user_token;
DROP TABLE user_detail;
DROP TABLE user_signin;
```

```
#Account
CREATE TABLE `user_signin` (
  `userid` INT(11) NOT NULL AUTO_INCREMENT UNIQUE PRIMARY KEY,
  `account_type` VARCHAR(14) NOT NULL,
  `email` VARCHAR(300) NOT NULL UNIQUE,
  `password` CHAR(60) NOT NULL UNIQUE,
  `isSignedIn` TINYINT(1) NOT NULL,
  `created_on` DATETIME NOT NULL,
  `updated_on` DATETIME
) ENGINE = InnoDB;

#Account
CREATE TABLE `user_detail` (
  `userid` INT(11) NOT NULL AUTO_INCREMENT UNIQUE,
  `firstname` VARCHAR(70) NOT NULL,
  `surname` VARCHAR(70) NOT NULL,
  `gender` VARCHAR(6) NOT NULL,
  `nationality` VARCHAR(70),
  `studentno` INT(9) NOT NULL UNIQUE,
  `degree` VARCHAR(70),
  `dateofbirth` DATE,
  `phonenumber` VARCHAR(70),
  `address1` VARCHAR(70),
  `address2` VARCHAR(70),
  `town` VARCHAR(70),
  `city` VARCHAR(70),
  `country` VARCHAR(70),
  `postcode` VARCHAR(70),
  `user_status` VARCHAR(9) NOT NULL,
  `created_on` DATETIME NOT NULL,
  `updated_on` DATETIME,
FOREIGN KEY (userid)
REFERENCES user_signin(userid)
ON UPDATE CASCADE
ON DELETE CASCADE
) ENGINE = InnoDB;

#Account
CREATE TABLE `user_token` (
  `userid` INT(11) NOT NULL AUTO_INCREMENT UNIQUE,
  `token` CHAR(60) UNIQUE,
  `created_on` DATETIME,
FOREIGN KEY (userid)
REFERENCES user_signin(userid)
ON UPDATE CASCADE
ON DELETE CASCADE
) ENGINE = InnoDB;

#Account
CREATE TABLE `user_fee` (
  `userid` INT(11) NOT NULL AUTO_INCREMENT UNIQUE,
  `fee_amount` NUMERIC(15,2) NOT NULL,
  `isHalf` TINYINT(1) NOT NULL,
  `created_on` DATETIME NOT NULL,
  `updated_on` DATETIME,
  FOREIGN KEY (userid)
  REFERENCES user_signin(userid)
    ON UPDATE CASCADE
    ON DELETE CASCADE
) ENGINE = InnoDB;
```

```
#Account
CREATE TABLE `paypal_log` (
 `userid` INT(11) NOT NULL,
 `paymentid` INT(11) NOT NULL AUTO_INCREMENT UNIQUE PRIMARY KEY,
 `invoiceid` BIGINT(10) NOT NULL UNIQUE,
 `transactionid` VARCHAR(17) NOT NULL UNIQUE,
 `productid` INT(1) NOT NULL,
 `product_name` VARCHAR(70) NOT NULL,
 `product_quantity` INT(11) NOT NULL,
 `product_amount` NUMERIC(15,2) NOT NULL,
 `payer_firstname` VARCHAR(70) NOT NULL,
 `payer_surname` VARCHAR(70) NOT NULL,
 `payer_email` VARCHAR(300) NOT NULL,
 `payer_phonenumber` VARCHAR(70),
 `payer_address1` VARCHAR(70) NOT NULL,
 `payer_address2` VARCHAR(70),
 `payer_town` VARCHAR(70),
 `payer_city` VARCHAR(70) NOT NULL,
 `payer_country` VARCHAR(70) NOT NULL,
 `payer_postcode` VARCHAR(70) NOT NULL,
 `payment_type` VARCHAR(5) NOT NULL,
 `payment_status` VARCHAR(9) NOT NULL,
 `created_on` DATETIME NOT NULL,
 `updated_on` DATETIME,
 `completed_on` DATETIME,
 `cancelled_on` DATETIME,
FOREIGN KEY (userid)
REFERENCES user_signin(userid)
ON UPDATE CASCADE
ON DELETE CASCADE
) ENGINE = InnoDB;

#Messenger
CREATE TABLE `user_message` (
 `messageid` INT(11) NOT NULL AUTO_INCREMENT UNIQUE PRIMARY KEY,
 `message_subject` VARCHAR(300) NOT NULL,
 `message_body` VARCHAR(10000) NOT NULL,
  `message_status` VARCHAR(9) NOT NULL,
  `created_on` DATETIME NOT NULL
) ENGINE = InnoDB;

#Messenger
CREATE TABLE `user_message_sent` (
  `messageid` INT(11) NOT NULL AUTO_INCREMENT,
  `message_from` INT(11) NOT NULL,
  `message_to` INT(11) NOT NULL,
FOREIGN KEY (messageid)
REFERENCES user_message(messageid),
FOREIGN KEY (message_from)
REFERENCES user_signin(userid),
FOREIGN KEY (message_to)
REFERENCES user_signin(userid)
ON UPDATE CASCADE
ON DELETE CASCADE
) ENGINE = InnoDB;
```

```
#Messenger
CREATE TABLE `user_message_received` (
  `messageid` INT(11) NOT NULL AUTO_INCREMENT,
  `message_from` INT(11) NOT NULL,
  `message_to` INT(11) NOT NULL,
  `isRead` TINYINT(1) NOT NULL,
FOREIGN KEY (messageid)
REFERENCES user_message(messageid),
FOREIGN KEY (message_from)
REFERENCES user_signin(userid),
FOREIGN KEY (message_to)
REFERENCES user_signin(userid)
ON UPDATE CASCADE
ON DELETE CASCADE
) ENGINE = InnoDB;

#Events
CREATE TABLE `system_event` (
 `eventid` INT(11) NOT NULL AUTO_INCREMENT UNIQUE PRIMARY KEY,
 `event_class` VARCHAR(15) NOT NULL,
 `event_name` VARCHAR(300) NOT NULL,
 `event_notes` VARCHAR(10000),
 `event_url` VARCHAR(300),
 `event_from` DATETIME NOT NULL,
 `event_to` DATETIME NOT NULL,
 `event_amount` NUMERIC(15,2) NOT NULL,
 `event_ticket_no` INT(11) NOT NULL,
 `event_status` VARCHAR(10) NOT NULL,
 `created_on` DATETIME NOT NULL,
 `updated_on` DATETIME
) ENGINE = InnoDB;

#Events
CREATE TABLE `system_event_booked` (
  `userid` INT(11) NOT NULL,
  `eventid` INT(11) NOT NULL,
  `event_class` VARCHAR(15) NOT NULL,
  `event_amount_paid` NUMERIC(15,2) NOT NULL,
  `ticket_quantity` INT(11) NOT NULL,
  `booked_on` DATETIME NOT NULL,
FOREIGN KEY (userid)
REFERENCES user_signin(userid),
FOREIGN KEY (eventid)
REFERENCES system_event(eventid)
ON UPDATE CASCADE
ON DELETE CASCADE
) ENGINE = InnoDB;

#University Map
CREATE TABLE `system_map_marker` (
  `markerid` INT(11) NOT NULL AUTO_INCREMENT UNIQUE PRIMARY KEY,
  `marker_name` VARCHAR(70) NOT NULL,
  `marker_notes` VARCHAR(10000),
  `marker_url` VARCHAR(300),
  `marker_lat` FLOAT(10,6) NOT NULL,
  `marker_long` FLOAT(10,6) NOT NULL,
  `marker_category` VARCHAR(70) NOT NULL,
  `marker_status` VARCHAR(9) NOT NULL,
  `created_on` DATETIME NOT NULL,
  `updated_on` DATETIME
) ENGINE = InnoDB;
```

```
#Calendar
CREATE TABLE `user_task` (
  `userid` INT(11) NOT NULL,
  `taskid` INT(11) NOT NULL AUTO_INCREMENT UNIQUE PRIMARY KEY,
  `task_name` VARCHAR(70) NOT NULL,
  `task_notes` VARCHAR(10000),
  `task_url` VARCHAR(300),
  `task_class` VARCHAR(15) NOT NULL,
  `task_startdate` DATETIME NOT NULL,
  `task_duedate` DATETIME NOT NULL,
  `task_status` VARCHAR(10) NOT NULL,
  `created_on` DATETIME NOT NULL,
  `updated_on` DATETIME,
  `completed_on` DATETIME,
FOREIGN KEY (userid)
REFERENCES user_signin(userid)
ON UPDATE CASCADE
ON DELETE CASCADE
) ENGINE = InnoDB;

#Library
CREATE TABLE `system_book` (
 `bookid` INT(11) NOT NULL AUTO_INCREMENT UNIQUE PRIMARY KEY,
 `book_name` VARCHAR(300) NOT NULL,
  `book_author` VARCHAR(300) NOT NULL,
  `book_notes` VARCHAR(10000),
  `book_copy_no` INT(11) NOT NULL,
  `book_location` VARCHAR(300) NOT NULL,
  `book_publisher` VARCHAR(300) NOT NULL,
  `book_publish_date` DATE NOT NULL,
  `book_publish_place` VARCHAR(300) NOT NULL,
  `book_page_amount` INT(11) NOT NULL,
  `book_barcode` INT(11) NOT NULL,
  `book_discipline` VARCHAR(300) NOT NULL,
  `book_language` VARCHAR(70) NOT NULL,
 `book_status` VARCHAR(9) NOT NULL,
  `isReserved` TINYINT(1) NOT NULL,
  `isLoaned` TINYINT(1) NOT NULL,
  `isRequested` TINYINT(1) NOT NULL,
 `created_on` DATETIME NOT NULL,
 `updated_on` DATETIME
) ENGINE = InnoDB;

#Library
CREATE TABLE `system_book_reserved` (
  `userid` INT(11) NOT NULL,
  `bookid` INT(11) NOT NULL,
  `reservationid` INT(11) NOT NULL AUTO_INCREMENT UNIQUE PRIMARY KEY,
 `tocollect_on` DATE NOT NULL,
  `collected_on` DATE NOT NULL,
 `isCollected` TINYINT(1) NOT NULL,
  `reservation_status` VARCHAR(9) NOT NULL,
  `created_on` DATETIME NOT NULL,
  `updated_on` DATETIME,
FOREIGN KEY (userid)
REFERENCES user_signin(userid),
FOREIGN KEY (bookid)
REFERENCES system_book(bookid)
ON UPDATE CASCADE
ON DELETE CASCADE
) ENGINE = InnoDB;
```

```
#Library
CREATE TABLE `system_book_loaned` (
  `userid` INT(11) NOT NULL,
  `bookid` INT(11) NOT NULL,
  `loanid` INT(11) NOT NULL AUTO_INCREMENT UNIQUE PRIMARY KEY,
  `book_class` VARCHAR(15) NOT NULL,
  `toreturn_on` DATE NOT NULL,
  `returned_on` DATE NOT NULL,
  `isReturned` TINYINT(1) NOT NULL,
  `isRequested` TINYINT(1) NOT NULL,
  `loan_status` VARCHAR(9) NOT NULL,
  `created_on` DATETIME NOT NULL,
  `updated_on` DATETIME,
FOREIGN KEY (userid)
REFERENCES user_signin(userid),
FOREIGN KEY (bookid)
REFERENCES system_book(bookid)
ON UPDATE CASCADE
ON DELETE CASCADE
) ENGINE = InnoDB;

#Library
CREATE TABLE `system_book_requested` (
  `userid` INT(11) NOT NULL,
  `bookid` INT(11) NOT NULL,
  `requestid` INT(11) NOT NULL AUTO_INCREMENT UNIQUE PRIMARY KEY,
  `isRead` TINYINT(1) NOT NULL,
  `isApproved` TINYINT(1) NOT NULL,
  `request_status` VARCHAR(9) NOT NULL,
  `created_on` DATETIME NOT NULL,
  `updated_on` DATETIME,
FOREIGN KEY (userid)
REFERENCES user_signin(userid),
FOREIGN KEY (bookid)
REFERENCES system_book(bookid)
ON UPDATE CASCADE
ON DELETE CASCADE
) ENGINE = InnoDB;

#Transport
CREATE TABLE `tube_line_status_now` (
  `statusid` INT(11) NOT NULL AUTO_INCREMENT UNIQUE PRIMARY KEY,
  `tube_lineid` INT(11) NOT NULL UNIQUE,
  `tube_line` VARCHAR (70) NOT NULL,
  `tube_line_status` VARCHAR (70),
  `tube_line_info` VARCHAR(10000),
  `updated_on` DATETIME NOT NULL
) ENGINE = InnoDB;

#Transport
CREATE TABLE `tube_station_status_now` (
  `statusid` INT(11) NOT NULL AUTO_INCREMENT UNIQUE PRIMARY KEY,
  `tube_stationid` INT(11) NOT NULL UNIQUE,
  `tube_station` VARCHAR (70) NOT NULL,
  `tube_station_status` VARCHAR (70),
  `tube_station_info` VARCHAR(10000),
  `updated_on` DATETIME NOT NULL
) ENGINE = InnoDB;
```

```
#Transport
CREATE TABLE `tube_line_status_this_weekend` (
  `statusid` INT(11) NOT NULL AUTO_INCREMENT UNIQUE PRIMARY KEY,
  `tube_line` VARCHAR (70) NOT NULL,
  `tube_line_status` VARCHAR (70),
  `tube_line_info` VARCHAR(10000),
  `updated_on` DATETIME NOT NULL
) ENGINE = InnoDB;

#Transport
CREATE TABLE `tube_station_status_this_weekend` (
  `statusid` INT(11) NOT NULL AUTO_INCREMENT UNIQUE PRIMARY KEY,
  `tube_station` VARCHAR (70) NOT NULL,
  `tube_station_status` VARCHAR (70),
  `tube_station_info` VARCHAR(10000),
  `updated_on` DATETIME NOT NULL
) ENGINE = InnoDB;

#Transport
CREATE TABLE `cycle_hire_status_now` (
  `statusid` INT NOT NULL AUTO_INCREMENT UNIQUE PRIMARY KEY,
  `dockid` INT NOT NULL UNIQUE,
  `dock_name` VARCHAR (70) NOT NULL,
  `dock_installed` VARCHAR (5),
  `dock_locked` VARCHAR(5),
  `dock_temporary` VARCHAR(5),
  `dock_bikes_available` INT(11),
  `dock_empty_docks` INT(11),
  `dock_total_docks` INT(11),
  `updated_on` DATETIME NOT NULL
) ENGINE = InnoDB;

#Timetable
CREATE TABLE `system_module` (
 `moduleid` INT(11) NOT NULL AUTO_INCREMENT UNIQUE PRIMARY KEY,
 `module_name` VARCHAR(300) NOT NULL,
 `module_notes` VARCHAR(10000),
 `module_url` VARCHAR(300),
 `module_status` VARCHAR(10) NOT NULL,
 `created_on` DATETIME NOT NULL,
 `updated_on` DATETIME
) ENGINE = InnoDB;

#Timetable
CREATE TABLE `user_module` (
  `userid` INT(11) NOT NULL,
  `moduleid` INT(11) NOT NULL,
FOREIGN KEY (userid)
REFERENCES user_signin(userid),
FOREIGN KEY (moduleid)
REFERENCES system_module(moduleid)
ON UPDATE CASCADE
ON DELETE CASCADE
) ENGINE = InnoDB;
```

```
#Timetable
CREATE TABLE `system_lecture` (
 `moduleid` INT(11) NOT NULL,
 `lectureid` INT(11) NOT NULL AUTO_INCREMENT UNIQUE PRIMARY KEY,
 `lecture_name` VARCHAR(300) NOT NULL,
 `lecture_lecturer` INT(11) NOT NULL,
 `lecture_notes` VARCHAR(10000),
 `lecture_day` VARCHAR(9) NOT NULL,
 `lecture_from_time` TIME NOT NULL,
 `lecture_to_time` TIME NOT NULL,
 `lecture_from_date` DATE NOT NULL,
 `lecture_to_date` DATE NOT NULL,
 `lecture_location` VARCHAR(300) NOT NULL,
 `lecture_capacity` INT(11) NOT NULL,
 `lecture_status` VARCHAR(9) NOT NULL,
 `created_on` DATETIME NOT NULL,
 `updated_on` DATETIME,
FOREIGN KEY (moduleid)
REFERENCES system_module(moduleid)
ON UPDATE CASCADE
ON DELETE CASCADE
) ENGINE = InnoDB;

#Timetable
CREATE TABLE `user_lecture` (
  `userid` INT(11) NOT NULL,
  `lectureid` INT(11) NOT NULL,
FOREIGN KEY (userid)
REFERENCES user_signin(userid),
FOREIGN KEY (lectureid)
REFERENCES system_lecture(lectureid)
ON UPDATE CASCADE
ON DELETE CASCADE
) ENGINE = InnoDB;

#Timetable
CREATE TABLE `system_tutorial` (
 `moduleid` INT(11) NOT NULL,
 `tutorialid` INT(11) NOT NULL AUTO_INCREMENT UNIQUE PRIMARY KEY,
 `tutorial_name` VARCHAR(300) NOT NULL,
 `tutorial_assistant` INT(11) NOT NULL,
 `tutorial_notes` VARCHAR(10000),
 `tutorial_day` VARCHAR(9) NOT NULL,
 `tutorial_from_time` TIME NOT NULL,
 `tutorial_to_time` TIME NOT NULL,
 `tutorial_from_date` DATE NOT NULL,
 `tutorial_to_date` DATE NOT NULL,
 `tutorial_location` VARCHAR(300) NOT NULL,
 `tutorial_capacity` INT(11) NOT NULL,
 `tutorial_status` VARCHAR(9) NOT NULL,
 `created_on` DATETIME NOT NULL,
 `updated_on` DATETIME,
FOREIGN KEY (moduleid)
REFERENCES system_module(moduleid)
ON UPDATE CASCADE
ON DELETE CASCADE
) ENGINE = InnoDB;
```

```
#Timetable
CREATE TABLE `user_tutorial` (
  `userid` INT(11) NOT NULL,
  `tutorialid` INT(11) NOT NULL,
FOREIGN KEY (userid)
REFERENCES user_signin(userid),
FOREIGN KEY (tutorialid)
REFERENCES system_tutorial(tutorialid)
ON UPDATE CASCADE
ON DELETE CASCADE
) ENGINE = InnoDB;

#Timetable
CREATE TABLE `system_exam` (
 `moduleid` INT(11) NOT NULL,
 `examid` INT(11) NOT NULL AUTO_INCREMENT UNIQUE PRIMARY KEY,
 `exam_name` VARCHAR(300) NOT NULL,
 `exam_notes` VARCHAR(10000),
 `exam_date` DATE NOT NULL,
 `exam_time` TIME NOT NULL,
 `exam_location` VARCHAR(300) NOT NULL,
 `exam_capacity` INT(11) NOT NULL,
 `exam_status` VARCHAR(9) NOT NULL,
 `created_on` DATETIME NOT NULL,
 `updated_on` DATETIME,
FOREIGN KEY (moduleid)
REFERENCES system_module(moduleid)
ON UPDATE CASCADE
ON DELETE CASCADE
) ENGINE = InnoDB;

#Timetable
CREATE TABLE `user_exam` (
  `userid` INT(11) NOT NULL,
  `examid` INT(11) NOT NULL,
FOREIGN KEY (userid)
REFERENCES user_signin(userid),
FOREIGN KEY (examid)
REFERENCES system_exam(examid)
ON UPDATE CASCADE
ON DELETE CASCADE
) ENGINE = InnoDB;
```

```
#Results
CREATE TABLE `user_result` (
  `userid` INT(11) NOT NULL,
  `moduleid` INT(11) NOT NULL,
  `resultid` INT(11) NOT NULL AUTO_INCREMENT UNIQUE PRIMARY KEY,
  `result_coursework_mark` NUMERIC(15,2),
  `result_exam_mark` NUMERIC(15,2),
  `result_overall_mark` NUMERIC(15,2),
  `result_notes` VARCHAR(10000),
  `result_status` VARCHAR(9) NOT NULL,
  `created_on` DATETIME NOT NULL,
  `updated_on` DATETIME,
FOREIGN KEY (userid)
REFERENCES user_signin(userid),
FOREIGN KEY (moduleid)
REFERENCES system_module(moduleid)
ON UPDATE CASCADE
ON DELETE CASCADE
) ENGINE = InnoDB;

#Feedback
CREATE TABLE `user_feedback` (
  `moduleid` INT(11) NOT NULL,
  `feedbackid` INT(11) NOT NULL AUTO_INCREMENT UNIQUE PRIMARY KEY,
  `feedback_subject` VARCHAR(300) NOT NULL,
  `feedback_body` VARCHAR(10000) NOT NULL,
  `isApproved` TINYINT(1) NOT NULL,
  `feedback_status` VARCHAR(9) NOT NULL,
  `created_on` DATETIME NOT NULL,
FOREIGN KEY (moduleid)
REFERENCES system_module(moduleid)
) ENGINE = InnoDB;

#Feedback
CREATE TABLE `user_feedback_sent` (
  `feedbackid` INT(11) NOT NULL,
  `feedback_from` INT(11) NOT NULL,
  `moduleid` INT(11) NOT NULL,
  `module_staff` INT(11) NOT NULL,
FOREIGN KEY (feedbackid)
REFERENCES user_feedback(feedbackid),
FOREIGN KEY (feedback_from)
REFERENCES user_signin(userid),
FOREIGN KEY (moduleid)
REFERENCES system_module(moduleid),
FOREIGN KEY (module_staff)
REFERENCES user_signin(userid)
ON UPDATE CASCADE
ON DELETE CASCADE
) ENGINE = InnoDB;
```

```
#Feedback
CREATE TABLE `user_feedback_received` (
  `feedbackid` INT(11) NOT NULL,
  `feedback_from` INT(11) NOT NULL,
  `moduleid` INT(11) NOT NULL,
  `module_staff` INT(11) NOT NULL,
  `isRead` TINYINT(1) NOT NULL,
FOREIGN KEY (feedbackid)
REFERENCES user_feedback(feedbackid),
FOREIGN KEY (feedback_from)
REFERENCES user_signin(userid),
FOREIGN KEY (moduleid)
REFERENCES system_module(moduleid),
FOREIGN KEY (module_staff)
REFERENCES user_signin(userid)
ON UPDATE CASCADE
ON DELETE CASCADE
) ENGINE = InnoDB;
```

DELETE Statements

```
DROP TABLE user_feedback_sent;
DROP TABLE user_feedback_received;
DROP TABLE user_feedback;

DROP TABLE user_exam;
DROP TABLE system_exam;

DROP TABLE user_result;

DROP TABLE user_lecture;
DROP TABLE system_lecture;
DROP TABLE user_tutorial;
DROP TABLE system_tutorial;
DROP TABLE user_module;
DROP TABLE system_module;

DROP TABLE cycle_hire_status_now;
DROP TABLE tube_station_status_this_weekend;
DROP TABLE tube_line_status_this_weekend;
DROP TABLE tube_station_status_now;
DROP TABLE tube_line_status_now;

DROP TABLE system_book_reserved;
DROP TABLE system_book_loaned;
DROP TABLE system_book_requested;
DROP TABLE system_book;

DROP TABLE user_task;

DROP TABLE system_map_marker;

DROP TABLE system_event_booked;
DROP TABLE system_event;


DROP TABLE user_message_sent;
DROP TABLE user_message_received;
DROP TABLE user_message;

DROP TABLE paypal_log;
DROP TABLE user_fee;
DROP TABLE user_token;
DROP TABLE user_detail;
DROP TABLE user_signin;
```

user_signin

```
CREATE TABLE `student_portal`.`user_signin` (
  `userid` INT(11) NOT NULL AUTO_INCREMENT UNIQUE PRIMARY KEY,
  `account_type` VARCHAR(14) NOT NULL,
  `email` VARCHAR(300) NOT NULL UNIQUE,
  `password` CHAR(60) NOT NULL UNIQUE,
  `isSignedIn` TINYINT(1) NOT NULL,
  `created_on` DATETIME NOT NULL,
  `updated_on` DATETIME
) ENGINE = InnoDB;
```

user_detail

```
CREATE TABLE `student_portal`.`user_detail` (
  `userid` INT(11) NOT NULL AUTO_INCREMENT UNIQUE,
  `firstname` VARCHAR(70) NOT NULL,
  `surname` VARCHAR(70) NOT NULL,
  `gender` VARCHAR(6) NOT NULL,
  `nationality` VARCHAR(70),
  `studentno` INT(9) NOT NULL UNIQUE,
  `degree` VARCHAR(70),
  `dateofbirth` DATE,
  `phonenumber` VARCHAR(70),
  `address1` VARCHAR(70),
  `address2` VARCHAR(70),
  `town` VARCHAR(70),
  `city` VARCHAR(70),
  `country` VARCHAR(70),
  `postcode` VARCHAR(70),
  `user_status` VARCHAR(9) NOT NULL,
  `created_on` DATETIME NOT NULL,
  `updated_on` DATETIME,
FOREIGN KEY (userid)
REFERENCES user_signin(userid)
ON UPDATE CASCADE
ON DELETE CASCADE
) ENGINE = InnoDB;
```

user_token

```
CREATE TABLE `student_portal`.`user_token` (
  `userid` INT(11) NOT NULL AUTO_INCREMENT UNIQUE,
  `token` CHAR(60) UNIQUE,
  `created_on` DATETIME,
FOREIGN KEY (userid)
REFERENCES user_signin(userid)
ON UPDATE CASCADE
ON DELETE CASCADE
) ENGINE = InnoDB;
```

user_fee

```
CREATE TABLE `student_portal`.`user_fees` (
  `userid` INT(11) NOT NULL AUTO_INCREMENT UNIQUE,
  `fee_amount` NUMERIC(15,2) NOT NULL,
  `isHalf` TINYINT(1) NOT NULL,
  `created_on` DATETIME NOT NULL,
  `updated_on` DATETIME,
FOREIGN KEY (userid)
REFERENCES user_signin(userid)
ON UPDATE CASCADE
ON DELETE CASCADE
) ENGINE = InnoDB;
```

paypal_log

```
CREATE TABLE `student_portal`.`paypal_log` (
  `userid` INT(11) NOT NULL,
  `paymentid` INT(11) NOT NULL AUTO_INCREMENT UNIQUE,
  `invoiceid` BIGINT(10) NOT NULL UNIQUE,
  `transactionid` VARCHAR(17) NOT NULL UNIQUE,
  `productid` INT(1) NOT NULL UNIQUE,
  `product_name` VARCHAR(70) NOT NULL,
  `product_quantity` INT(11) NOT NULL,
  `product_amount` NUMERIC(15,2) NOT NULL,
  `payer_firstname` VARCHAR(70) NOT NULL,
  `payer_surname` VARCHAR(70) NOT NULL,
  `payer_email` VARCHAR(300) NOT NULL,
  `payer_phonenumber` VARCHAR(70),
  `payer_address1` VARCHAR(70) NOT NULL,
  `payer_address2` VARCHAR(70),
  `payer_town` VARCHAR(70),
  `payer_city` VARCHAR(70) NOT NULL,
  `payer_country` VARCHAR(70) NOT NULL,
  `payer_postcode` VARCHAR(70) NOT NULL,
  `payment_type` VARCHAR(5) NOT NULL,
  `payment_status` VARCHAR(9) NOT NULL,
  `created_on` DATETIME NOT NULL,
  `updated_on` DATETIME,
  `completed_on` DATETIME,
  `cancelled_on` DATETIME,
FOREIGN KEY (userid)
REFERENCES user_signin(userid)
ON UPDATE CASCADE
ON DELETE CASCADE
) ENGINE = InnoDB;
```

user_message

```
CREATE TABLE `student_portal`.`user_message` (
  `messageid` INT(11) NOT NULL AUTO_INCREMENT UNIQUE PRIMARY KEY,
  `message_subject` VARCHAR(300) NOT NULL,
  `message_body` VARCHAR(5000),
  `message_status` VARCHAR(9),
  `created_on` DATETIME NOT NULL
) ENGINE = InnoDB;
```

user_message_sent

```
CREATE TABLE `student_portal`.`user_messages_sent` (
  `messageid` INT(11) NOT NULL AUTO_INCREMENT,
  `message_from` INT(11) NOT NULL,
  `message_to` INT(11) NOT NULL,
FOREIGN KEY (messageid)
REFERENCES user_message(messageid),
FOREIGN KEY (message_from)
REFERENCES user_signin(userid),
FOREIGN KEY (message_to)
REFERENCES user_signin(userid)
ON UPDATE CASCADE
ON DELETE CASCADE
) ENGINE = InnoDB;
```

user_message_received

```
CREATE TABLE `student_portal`.`user_message_received` (
  `messageid` INT(11) NOT NULL AUTO_INCREMENT,
  `message_from` INT(11) NOT NULL,
  `message_to` INT(11) NOT NULL,
  `isRead` TINYINT(1) NOT NULL,
FOREIGN KEY (messageid)
REFERENCES user_message(messageid),
FOREIGN KEY (message_from)
REFERENCES user_signin(userid),
FOREIGN KEY (message_to)
REFERENCES user_signin(userid)
ON UPDATE CASCADE
ON DELETE CASCADE
) ENGINE = InnoDB;
```

system_event

```
CREATE TABLE `student_portal`.`system_event` (
  `eventid` INT(11) NOT NULL AUTO_INCREMENT UNIQUE PRIMARY KEY,
  `event_class` VARCHAR(15) NOT NULL,
  `event_name` VARCHAR(300) NOT NULL,
  `event_notes` VARCHAR(10000),
  `event_url` VARCHAR(300),
  `event_from` DATETIME NOT NULL,
  `event_to` DATETIME NOT NULL,
  `event_amount` NUMERIC(15,2) NOT NULL,
  `event_ticket_no` INT(11) NOT NULL,
  `event_status` VARCHAR(10) NOT NULL,
  `created_on` DATETIME NOT NULL,
  `updated_on` DATETIME
) ENGINE = InnoDB;
```

system_event_booked

```
CREATE TABLE `student_portal`.`system_event_booked` (
  `userid` INT(11) NOT NULL,
  `eventid` INT(11) NOT NULL,
  `event_class` VARCHAR(15) NOT NULL,
  `event_amount` NUMERIC(15,2) NOT NULL,
  `ticket_quantity` VARCHAR(11) NOT NULL,
  `booked_on` DATETIME NOT NULL,
FOREIGN KEY (userid)
REFERENCES user_signin(userid),
FOREIGN KEY (eventid)
REFERENCES system_event(eventid)
ON UPDATE CASCADE
ON DELETE CASCADE
) ENGINE = InnoDB;
```

system_map_marker

```
CREATE TABLE `system_map_marker` (
  `markerid` INT(11) NOT NULL AUTO_INCREMENT UNIQUE PRIMARY KEY,
  `marker_name` VARCHAR (70) NOT NULL,
  `marker_notes` VARCHAR (10000),
  `marker_url` VARCHAR(300),
  `marker_lat` FLOAT(10,6) NOT NULL,
  `marker_long` FLOAT(10,6) NOT NULL,
  `marker_category` VARCHAR (70) NOT NULL
  `marker_status` VARCHAR(9) NOT NULL,
  `created_on` DATETIME NOT NULL,
  `updated_on` DATETIME
) ENGINE = InnoDB;
```

user_task

```
CREATE TABLE `student_portal`.`user_task` (
  `userid` INT(11) NOT NULL,
  `taskid` INT(11) NOT NULL AUTO_INCREMENT UNIQUE PRIMARY KEY,
  `task_name` VARCHAR(70) NOT NULL,
  `task_notes` VARCHAR(10000),
  `task_url` VARCHAR(300),
  `task_class` VARCHAR(15) NOT NULL,
  `task_startdate` DATETIME NOT NULL,
  `task_duedate` DATETIME NOT NULL,
  `task_status` VARCHAR(10) NOT NULL,
  `created_on` DATETIME NOT NULL,
  `updated_on` DATETIME,
  `completed_on` DATETIME,
FOREIGN KEY (userid)
REFERENCES user_signin(userid)
ON UPDATE CASCADE
ON DELETE CASCADE
) ENGINE = InnoDB;
```

system_book

```
CREATE TABLE `student_portal`.`system_book` (
  `bookid` INT(11) NOT NULL AUTO_INCREMENT UNIQUE PRIMARY KEY,
  `book_name` VARCHAR(300) NOT NULL,
  `book_author` VARCHAR(300) NOT NULL,
  `book_notes` VARCHAR(10000),
  `book_copy_no` INT(11) NOT NULL,
  `book_location` VARCHAR(300) NOT NULL,
  `book_publisher` VARCHAR(300) NOT NULL,
  `book_publish_date` DATE NOT NULL,
  `book_publish_place` VARCHAR(300) NOT NULL,
  `book_page_amount` INT(11) NOT NULL,
  `book_barcode` INT(11) NOT NULL,
  `book_discipline` VARCHAR(300) NOT NULL,
  `book_language` VARCHAR(70) NOT NULL,
  `book_status` VARCHAR(9) NOT NULL,
  `isReserved` TINYINT(1) NOT NULL,
  `isLoaned` TINYINT(1) NOT NULL,
  `isRequested` TINYINT(1) NOT NULL,
  `created_on` DATETIME NOT NULL,
  `updated_on` DATETIME
) ENGINE = InnoDB;
```

system_book_reserved

```
CREATE TABLE `student_portal`.`system_book_reserved` (
  `userid` INT(11) NOT NULL,
  `bookid` INT(11) NOT NULL,
  `reservationid` INT(11) NOT NULL AUTO_INCREMENT UNIQUE PRIMARY KEY,
  `tocollect_on` DATE NOT NULL,
  `collected_on` DATE NOT NULL,
  `isCollected` TINYINT(1) NOT NULL,
  `reservation_status` VARCHAR(9) NOT NULL,
  `created_on` DATETIME NOT NULL,
  `updated_on` DATETIME,
FOREIGN KEY (userid)
REFERENCES user_signin(userid)
FOREIGN KEY (bookid)
REFERENCES system_book(bookid),
ON UPDATE CASCADE
ON DELETE CASCADE
) ENGINE = InnoDB;
```

system_book_loaned

```
CREATE TABLE `student_portal`.`system_book_reserved` (
  `userid` INT(11) NOT NULL,
  `bookid` INT(11) NOT NULL,
  `loanid` INT(11) NOT NULL AUTO_INCREMENT UNIQUE PRIMARY KEY,
  `book_class` VARCHAR(15) NOT NULL,
  `toreturn_on` DATE NOT NULL,
  `returned_on` DATE NOT NULL,
  `isReturned` TINYINT(1) NOT NULL,
  `isRequested` TINYINT(1) NOT NULL,
  `loan_status` VARCHAR(9) NOT NULL,
  `created_on` DATETIME NOT NULL,
  `updated_on` DATETIME,
FOREIGN KEY (userid)
REFERENCES user_signin(userid)
FOREIGN KEY (bookid)
REFERENCES system_book(bookid),
ON UPDATE CASCADE
ON DELETE CASCADE
) ENGINE = InnoDB;
```

system_book_requested

```
CREATE TABLE `student_portal`.`system_book_reserved` (
  `userid` INT(11) NOT NULL,
  `bookid` INT(11) NOT NULL,
  `requestid` INT(11) NOT NULL AUTO_INCREMENT UNIQUE PRIMARY KEY,
  `isRead` TINYINT(1) NOT NULL,
  `isApproved` TINYINT(1) NOT NULL,
  `request_status` VARCHAR(9) NOT NULL,
  `created_on` DATETIME NOT NULL,
  `updated_on` DATETIME,
FOREIGN KEY (userid)
REFERENCES user_signin(userid)
FOREIGN KEY (bookid)
REFERENCES system_book(bookid),
ON UPDATE CASCADE
ON DELETE CASCADE
) ENGINE = InnoDB;
```

tube_line_status_now

```
CREATE TABLE `student_portal`.`tube_line_status_now` (
  `statusid` INT(11) NOT NULL AUTO_INCREMENT UNIQUE PRIMARY KEY,
  `tube_lineid` INT(11) NOT NULL UNIQUE,
  `tube_line` VARCHAR (70) NOT NULL,
  `tube_line_status` VARCHAR (70) NOT NULL,
  `tube_line_info` VARCHAR(1000) NOT NULL,
  `updated_on` DATETIME NOT NULL
) ENGINE = InnoDB;
```

tube_station_status_now

```
CREATE TABLE `student_portal`.`tube_station_status_now` (
  `statusid` INT(11) NOT NULL AUTO_INCREMENT UNIQUE PRIMARY KEY,
  `tube_stationid` INT(11) NOT NULL UNIQUE,
  `tube_station` VARCHAR (70) NOT NULL,
  `tube_station_status` VARCHAR (70) NOT NULL,
  `tube_station_info` VARCHAR(10000) NOT NULL,
  `updated_on` DATETIME NOT NULL
) ENGINE = InnoDB;
```

tube_line_status_this_weekend

```
CREATE TABLE `student_portal`.`tube_line_status_this_weekend` (
  `statusid` INT(11) NOT NULL AUTO_INCREMENT UNIQUE PRIMARY KEY,
  `tube_line` VARCHAR (70) NOT NULL,
  `tube_line_status` VARCHAR (70) NOT NULL,
  `tube_line_info` VARCHAR(10000) NOT NULL,
  `updated_on` DATETIME NOT NULL
) ENGINE = InnoDB;
```

tube_station_status_this_weekend

```
CREATE TABLE `student_portal`.`tube_station_status_this_weekend` (
  `statusid` INT(11) NOT NULL AUTO_INCREMENT UNIQUE PRIMARY KEY,
  `tube_station` VARCHAR (70) NOT NULL,
  `tube_station_status` VARCHAR (70) NOT NULL,
  `tube_station_info` VARCHAR(1000) NOT NULL,
  `updated_on` DATETIME NOT NULL
) ENGINE = InnoDB;
```

system_module

```
CREATE TABLE `student_portal`.`system_module` (
  `moduleid` INT(11) NOT NULL AUTO_INCREMENT UNIQUE PRIMARY KEY,
  `module_name` VARCHAR(300) NOT NULL,
  `module_notes` VARCHAR(10000),
  `module_url` VARCHAR(300),
  `module_status` VARCHAR(10) NOT NULL,
  `created_on` DATETIME NOT NULL,
  `updated_on` DATETIME
) ENGINE = InnoDB;
```

user_module

```
CREATE TABLE `student_portal`.`user_module` (
  `userid` INT(11) NOT NULL,
  `moduleid` INT(11) NOT NULL,
FOREIGN KEY (userid)
REFERENCES user_signin(userid),
FOREIGN KEY (moduleid)
REFERENCES system_module(moduleid)
ON UPDATE CASCADE
ON DELETE CASCADE
) ENGINE = InnoDB;
```

system_lecture

```
CREATE TABLE `student_portal`.`system_lecture` (
  `moduleid` INT(11) NOT NULL UNIQUE,
  `lectureid` INT(11) NOT NULL AUTO_INCREMENT UNIQUE PRIMARY KEY,
  `lecture_name` VARCHAR(300) NOT NULL,
  `lecture_lecturer` INT(11) NOT NULL,
  `lecture_notes` VARCHAR(10000),
  `lecture_day` VARCHAR(9) NOT NULL,
  `lecture_from_time` TIME NOT NULL,
  `lecture_to_time` TIME NOT NULL,
  `lecture_from_date` DATE NOT NULL,
  `lecture_to_date` DATE NOT NULL,
  `lecture_location` VARCHAR(300) NOT NULL,
  `lecture_capacity` INT(11) NOT NULL,
  `lecture_status` VARCHAR(9) NOT NULL,
  `created_on` DATETIME NOT NULL,
  `updated_on` DATETIME,
FOREIGN KEY (moduleid)
REFERENCES system_module(moduleid)
ON UPDATE CASCADE
ON DELETE CASCADE
) ENGINE = InnoDB;
```

user_lecture

```
CREATE TABLE `student_portal`.`user_lecture` (
  `userid` INT(11) NOT NULL,
  `lectureid` INT(11) NOT NULL,
FOREIGN KEY (userid)
REFERENCES user_signin(userid),
FOREIGN KEY (moduleid)
REFERENCES system_lecture(lectureid)
ON UPDATE CASCADE
ON DELETE CASCADE
) ENGINE = InnoDB;
```

system_tutorial

```
CREATE TABLE `student_portal`.`system_tutorial` (
  `moduleid` INT(11) NOT NULL UNIQUE,
  `tutorialid` INT(11) NOT NULL AUTO_INCREMENT UNIQUE PRIMARY KEY,
  `tutorial_name` VARCHAR(300) NOT NULL,
  `tutorial_assistant` INT(11) NOT NULL,
  `tutorial_notes` VARCHAR(10000),
  `tutorial_day` VARCHAR(9) NOT NULL,
  `tutorial_from_time` TIME NOT NULL,
  `tutorial_to_time` TIME NOT NULL,
  `tutorial_from_date` DATE NOT NULL,
  `tutorial_to_date` DATE NOT NULL,
  `tutorial_location` VARCHAR(300) NOT NULL,
  `tutorial_capacity` VARCHAR(11) NOT NULL,
  `tutorial_status` VARCHAR(9) NOT NULL,
  `created_on` DATETIME NOT NULL,
  `updated_on` DATETIME,
FOREIGN KEY (moduleid)
REFERENCES system_module(moduleid)
ON UPDATE CASCADE
ON DELETE CASCADE
) ENGINE = InnoDB;
```

user_tutorial

```
CREATE TABLE `student_portal`.`user_tutorial` (
  `userid` INT(11) NOT NULL,
  `tutorialid` INT(11) NOT NULL,
FOREIGN KEY (userid)
REFERENCES user_signin(userid),
FOREIGN KEY (moduleid)
REFERENCES system_tutorial(tutorialid)
ON UPDATE CASCADE
ON DELETE CASCADE
) ENGINE = InnoDB;
```

system_exam

```
CREATE TABLE `student_portal`.`system_exam` (
  `moduleid` INT(11) NOT NULL UNIQUE,
  `examid` INT(11) NOT NULL AUTO_INCREMENT UNIQUE PRIMARY KEY,
  `exam_name` VARCHAR(300) NOT NULL,
  `exam_notes` VARCHAR(10000),
  `exam_date` DATE NOT NULL,
  `exam_time` TIME NOT NULL,
  `exam_location` VARCHAR(300) NOT NULL,
  `exam_capacity` INT(11) NOT NULL,
  `exam_status` VARCHAR(9) NOT NULL,
  `created_on` DATETIME NOT NULL,
  `updated_on` DATETIME,
FOREIGN KEY (moduleid)
REFERENCES system_module(moduleid)
ON UPDATE CASCADE
ON DELETE CASCADE
) ENGINE = InnoDB;
```

user_exam

```
CREATE TABLE `student_portal`.`user_exam` (
  `userid` INT(11) NOT NULL,
  `examid` INT(11) NOT NULL,
FOREIGN KEY (userid)
REFERENCES user_signin(userid),
FOREIGN KEY (moduleid)
REFERENCES system_exam(examid)
ON UPDATE CASCADE
ON DELETE CASCADE
) ENGINE = InnoDB;
```

user_result

```
CREATE TABLE `student_portal`.`user_result` (
  `userid` INT(11) NOT NULL,
  `moduleid` INT(11) NOT NULL,
  `resultid` INT(11) NOT NULL AUTO_INCREMENT UNIQUE PRIMARY KEY,
  `result_coursework_mark` NUMERIC(15,2) NOT NULL,
  `result_exam_mark` NUMERIC(15,2) NOT NULL,
  `result_overall_mark` NUMERIC(15,2) NOT NULL,
  `result_notes` VARCHAR(10000) NOT NULL,
  `result_status` VARCHAR(9) NOT NULL,
  `created_on` DATETIME NOT NULL,
  `updated_on` DATETIME
FOREIGN KEY (userid)
REFERENCES user_signin(userid),
FOREIGN KEY (moduleid)
REFERENCES system_module (moduleid)
ON UPDATE CASCADE
ON DELETE CASCADE
) ENGINE = InnoDB;
```

user_feedback

```
CREATE TABLE `student_portal`.`user_feedback` (
  `moduleid` INT(11) NOT NULL,
  `feedbackid` INT(11) NOT NULL AUTO_INCREMENT UNIQUE PRIMARY KEY,
  `feedback_subject` VARCHAR(300) NOT NULL,
  `feedback_body` VARCHAR(10000) NOT NULL,
  `isApproved` TINYINT(1) NOT NULL,
  `feedback_status` VARCHAR(9) NOT NULL,
  `created_on` DATETIME NOT NULL
FOREIGN KEY (moduleid)
REFERENCES system_module(moduleid)
) ENGINE = InnoDB;
```

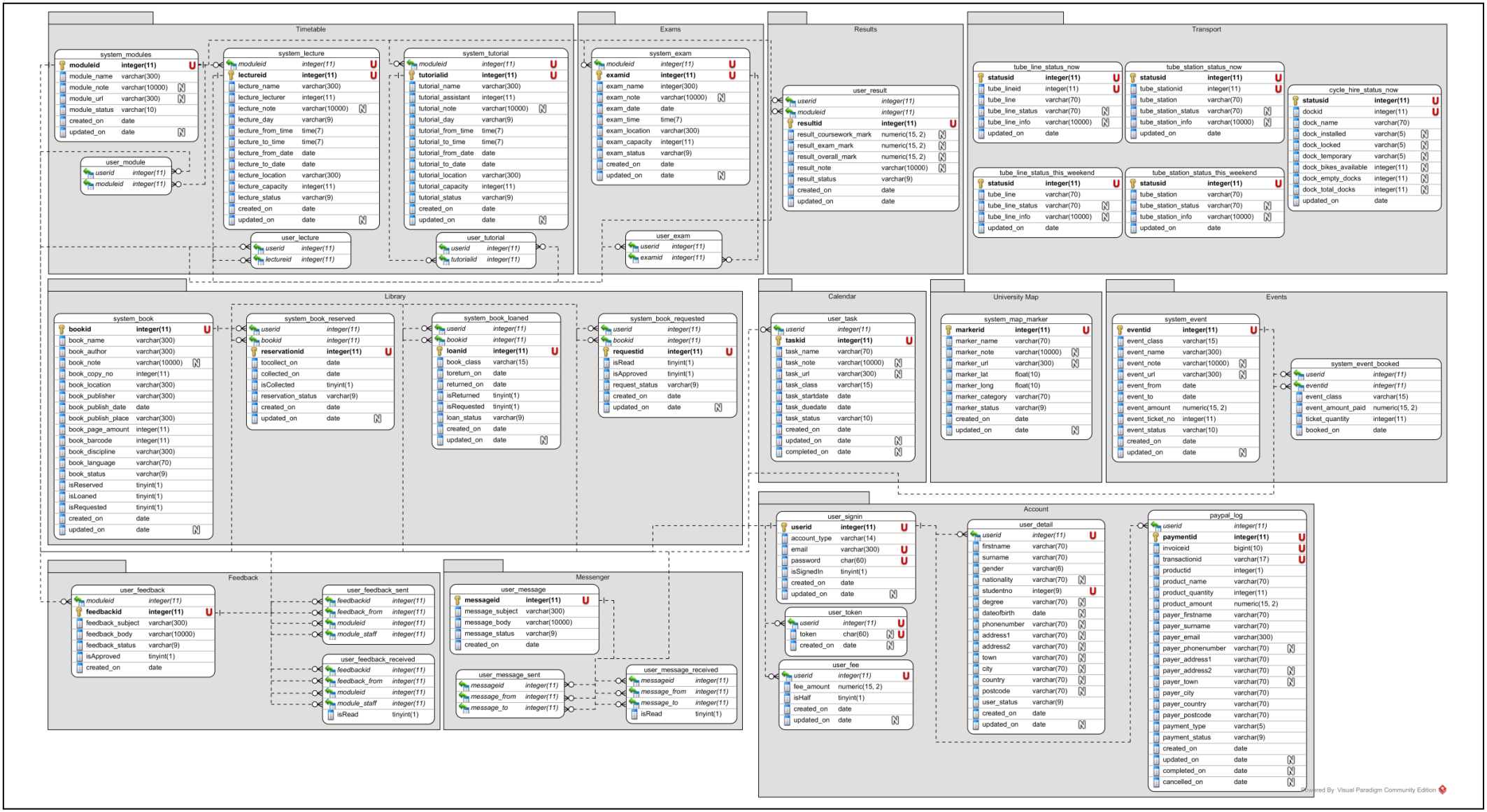
user_feedback_sent

```
CREATE TABLE `student_portal`.`user_feedback_sent` (
  `feedbackid` INT(11) NOT NULL,
  `feedback_from` INT(11) NOT NULL,
  `moduleid` INT(11) NOT NULL,
  `module_staff` INT(11) NOT NULL,
FOREIGN KEY (feedbackid)
REFERENCES user_feedback(feedbackid),
FOREIGN KEY (feedback_from)
REFERENCES user_signin(userid),
FOREIGN KEY (moduleid)
REFERENCES system_module(moduleid),
FOREIGN KEY (module_staff)
REFERENCES user_signin(userid)
ON UPDATE CASCADE
ON DELETE CASCADE
) ENGINE = InnoDB;
```

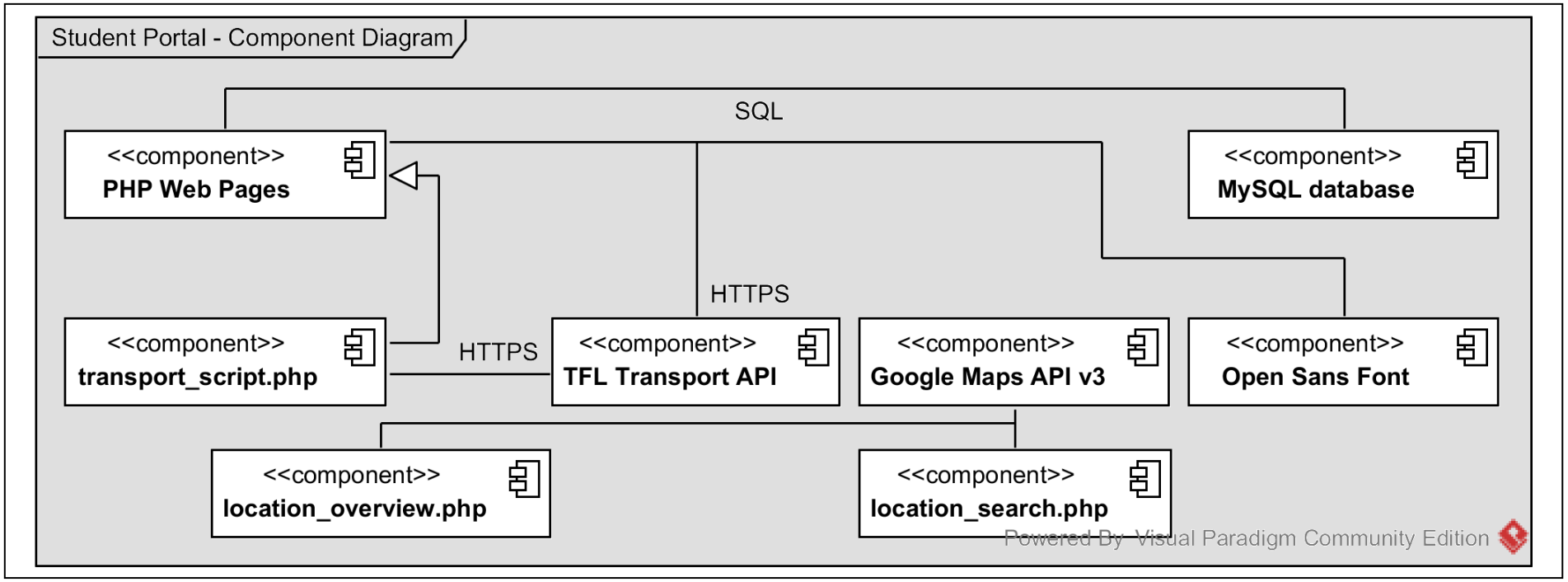
user_feedback_received

```
CREATE TABLE `student_portal`.`user_feedback_received` (
  `feedbackid` INT(11) NOT NULL,
  `feedback_from` INT(11) NOT NULL,
  `moduleid` INT(11) NOT NULL,
  `module_staff` INT(11) NOT NULL,
  `isRead` TINYINT(1) NOT NULL,
FOREIGN KEY (feedbackid)
REFERENCES user_feedback(feedbackid),
FOREIGN KEY (feedback_from)
REFERENCES user_signin(userid),
FOREIGN KEY (moduleid)
REFERENCES system_module(moduleid),
FOREIGN KEY (module_staff)
REFERENCES user_signin(userid)
ON UPDATE CASCADE
ON DELETE CASCADE
) ENGINE = InnoDB;
```

5. Entity Relationship diagram



6. Component diagram



7. Deployment diagram