Danmarks Tekniske Universitet

Project Management System - Part 1

Group 01
Aleksander Wind (s214683)
Alexander Bendix Gosden (s204209)
Nikolai Hansen (s214681)
William Peytz (s204145)

Software Engineering, 02161

Technical University of Denmark
April, 2022

Contents

Work distribution

	Aleksander W.	Alexander G.	Nikolai	William
Introduction:	0%	100%	0%	0%
Glossary:	10%	40%	10%	40%
Use case diagram:	40%	10%	40%	10%
Detailed use case:	20%	30%	20%	30%
Requirement Discussion:	0%	0%	0%	100%
Class Diagram:	10%	40%	40%	10%
Sequence Diagram:	Feature 1 & 2	Feature 7 & 8 & 9	Feature 3 & 4	Feature 5 & 6
Program Design Discussion:	0%	0%	0%	100%

1 Requirements

1.1 Introduction

This report features some of the preliminary work which was made in order to prepare for and make the development of the TimeManager software easier and more structured. This is the second version of this report, where some names, features and diagrams have been added or modified, in order to reflect our newfound knowledge that comes with developing the software, as well as to include the feedback from the first iteration.

The first part of the report is a glossary to keep track of the most important words we use, and their defintions, which is important for making diagrams and correct and consistent implementation. We have tried to make our glossary hierarchical such that the most important terms come first and more detailed and niche terms comes last. To create overview of all the terms in the defintion part of the glossary have been marked as bold.

Secondly a use case diagram of our program have been created, to get an overview of the different functionally of our software, and which also makes it easier to develop. Here 11 different use cases have been selected, which can be performed by either developers or project managers, as according to the diagram. Afterwards, in order to make sure our code works, 11 detailed use cases as cucumber features has been created, which tests our code for errors. For each of the features 1-5 scenarios have been created. These 11 detailed use cases are a continuation from the use case diagram for consistency reasons.
Then to wrap up the requirement section of the report there is a short discussion for some of the ambiguous project descriptions and how we handled these accordingly.

The second half of the report focuses on program design. Firstly by creating a class diagram of the program design. This has been updated throughout the development of the software. After the class diagram 9 out of the 11 detailed use cases have been selected and made into sequence diagrams. Note that these are just examples of sequence diagrams and methods for the different use cases. The actual use cases may have been cause to change and implemented differently with other methods. Some of the detailed use cases featured multiple scenarios. Multiple scenarios can be shown in one sequence diagram, but we decided to have different scenarios in different sequence diagrams for clarity purposes. This means that some use cases have several sequence diagrams. Finally, there will be a short discussion of some of the choices about the program design.

1.2 Glossary

See appendix A.

1.3 Use case diagram

See appendix B.

1.4 Detailed use cases

See appendix C.

1.5 Discussion

Due to the fact that the requirements for the project is a paragraph of text explaining the concept, and not an extensive list, it is natural for ambiguities in the project to arise. One example is that some of the features are not specifically mentioned, but implicitly needed to create the main functionalities. For example you need to be able to create developers before you can assign them to a project, and as such this functionality is needed. Some features only have a few requirements and the specifics are unspecified. An example of this is that there should be a feature to create reports, but there is no explicit mention of what should actually be in these reports other than expected work time left. Here we as developers have to make a decision on what we find most suitable. Another ambiguity we found in the project was that it sometimes be hard to tell who is authorized to do what. For example: Can a development employee create a project? How exactly is the project manager chosen? Who creates activities if there is no project manager on a project? Can the customer create activities? We have made the decision that project managers are developers who have been appointed, which means that project managers themselves have all the same abilities as developers. This means that in the use case diagram, the arrows from project manager should also be pointing to all uses cases, instead of only 4. However we decided to omit this from the use case diagram to make it easier to read. We decided that some abilities should be reserved for the project managers, as according to the use case diagram. We also decided that a project manager is project manager for a specific project and therefore only have their project manager permissions for the project they are project managers of. Some limitations have also been established for the project, in order to focus on the most essential features of the software. Two of these limitations include the decisions to omit to create a login system or a GUI, instead of a UI, as we concluded it was not nessercary for our implemention of the different usecases in the project.

2 Program Design

2.1 Class Diagram

See Appendix D.

2.2 Sequence Diagrams

See Appendix E.

2.3 Discussion of Program Design

When creating the class diagram and the sequence diagrams, we have generelly have had the attitude to keep all features as simple as possible, to avoid implementation problems and confusion. We have also tried to see all problems from a user perspective, in order to avoid creating unnecessary features or complicated design choices. We have also considered that all features must have the ability to be implemented with a simple UI.

3 Appendicies

3.1 Appendix A: Glossary

Term	Definition
TimeManager	The software which is used for **time registration** and **project management**.
Project	A **project** is a list of related **activities**, with a **project ID** and a **project name**. A **project** is either a **customer project** or an **internal project**. The **project** can be managed by a **project manager**. A **project** has a given **time budget** and a start and end week. A **project** is worked on by one or more **developers**.
Project ID	The ID of a **project**. This is a number, which is given by the **system** and has the structure of the year it's made and a running number. I.e. 22001 or 22002.
Project name	The name of a **project**.
Activity	**Activities** are subgoals of a **project** or an external activity such as vacation or a course. All **activities** have a name and a **time budget** rounded to the nearest half hour. The resolution of an **activity** is given in weeks where there is a start and end week. Both project activities and external activities can have one or more developers assigned to them
Time budget	The estimated number of hours for a **project** or **activity**.
Developer	An employee of Softwarehuset A/S which can be assigned to **projects** and **activities**. They each have an ID with up to four letters.
Project manager	A **project manager** is a chosen **developer**, which hands out **activities** to other **developers** of the **project** he/she is leading. A **project manager** is not necessarily chosen from the start of a **project**.
Customer	A **customer** can request a **project**. They prioritize **project planning** over **time registration** of **work hours** and **time registration** of work hours over **reports** for **project** meetings.
Customer project	A **project** which has been requested by a **customer**.
Internal project	A **project** which has been requested by a **developer**.
Time registration	A term for registering the **time spent** for a given **activity**.
Project planning	**Project planning** is the making of **activities** and setting **time budgets**, **starting time** for each **activity** for the **project**.
Time spent	The number of hours spent on an **activity**. The number is rounded to the nearest half hour.
Expected work time left	Estimated time left to complete the given **project**.
Free employees	A **developer** which can be assigned to an **activity**.
Request assistance	A feature which can be used by **developers** to add a **free employee** to work on a **activity** without being added as an developer on the project.
Report	A weekly text containing a summary of a given **project**'s status.

3.2 Appendix B: Use case diagram

Figure 1: Use case diagram

3.3 Appendix C: Detailed use cases

```
Feature: Create new project
  Description: Create either a new internal or customer project.
  Actors: Developer

  Scenario: Create a customer project
    Given that a project with name "NEM-ID" does not exist.
    When developer "whkp" creates a project with name "NEM-ID", as a "customer" project,
    with start week "2022-10" and end week "2022-13"
    Then the project with name "NEM-ID" is created.

  Scenario: Create a customer project with a name that already exists
    Given that a project with name "NEM-ID" already exists
    When developer "whkp" creates a project with name "NEM-ID", as a "customer" project,
    with start week "2022-10" and end week "2022-13"
    Then developer "whkp" is given the error message "Project could not be created,
    as the project name is already in use."


Feature: Create external activity
  Description: Create an external activity which is not linked to a project
  Actors: Developer

  Scenario: Create an external activity where all chosen developers are free employees
    Given that the developer "whkp" wants to create an external activity called "Software Course".
    When the developer "whkp" chooses to create an activity with name "Software Course"
    as an external activity, with start week "2022-15" and end week "2022-17".
    Then developer writes "pmka", "dskw" and "vmrr"
    Given that the chosen developers are free employees in the time period
    Then the external activity is created
    And the three developers are added to the external activity

   Scenario: Create external activity where a developer is not a free employee
     Given that the developer "whkp" wants to create an external activity called "Hardware Course".
     When the developer "whkp" chooses to create an activity with name "Hardware Course"
     as an external activity, with start week "2022-15" and end week "2022-17".
     Then developer writes "pmka", "dskw" and "vmrr"
     Given that at least one of the developers are not a free employee in the chosen time period
     Then developer "whkp" will be met with the error message "Developer is not free
     in the given time period."


Feature: Assign a project manager to a project
  Description: Assign a project manager to customer project
  Actors: Developer
  Scenario: A developer adds a project manager to a customer project
    Given that the "customer" project "NEM-ID" with start week "2022-15"
    and end week "2022-45" exists.
    And that the developer "asdg" and "whkp" exists
    Then the developer "asdg" assigns "wkhp" as project manager for project "NEM-ID"

  Scenario: A developer adds a new project manager to a customer project with an
    existing project manager
    Given that the "customer" project "NEM-ID" with start week "2022-15" and end week "2022-45" exists.
    And that the developer "asdg", "whkp" and "niha" exists
    And that "wkhp" is project manager for "customer" project "NEM-ID"
    When the developer "asdg" assigns "niha" as project manager for project "NEM-ID"
    Then the new project manager for "customer" project "NEM-ID" is "niha"
```

```
Feature: Request Assistance
  Description: A Developer requests assistance from another developer not assigned to
  Actors: Developer

  Scenario: Request Assistance Successfully
    Given that the "customer" project "NEM-ID" with start week "2022-09" and end week "2022-49" exists
    And that the developer "asbg" is assigned a project activity "Debugging" with start week "2022-40",
    end week "2022-42" and time budget 10 exists
    When "asbg" requests assistance for "Debugging" from developer "nikh"
    Then "nikh" will be able to register hours spent on the activity "Debugging"

  Scenario: Request Assistance from developer who is not available
    Given that the "customer" project "NEM-ID" with start week "2022-09" and end week "2022-49" exists
    And that the developer "asbg" is assigned a project activity "Debugging" with start week "2022-40",
    end week "2022-42" and time budget 10 exists
    When "asbg" requests assistance for "Debugging" from busy developer "nikh"
    Then the error "Developer is not free in the given time period."  is returned

  Scenario: Request Assistance from developer who is already on activity
    Given that the "customer" project "NEM-ID" with start week "2022-09" and end week "2022-49" exists
    And that the developers "asbg" and "nikh" are assigned a project activity "Debugging" with start
    week "2022-40", end week "2022-42" and time budget 10 exists
    When "asbg" requests assistance for "Debugging" from developer "nikh"
    Then the error "Developer is already working on the activity"  is returned

Feature: Register Time spent on activity
    Description: Developer records time spent on project activity on a given date. Actors: Developer

    Scenario: Register time for activity on a given date.
        Given that a "customer" project "NEM-ID" with start week "2022-04" and end
        week "2022-45" exists
        And the project activity "Debugging" with start week "2022-05" and end week
        "2022-07"
        and time budget 20 exists
And that the developer "alew" is assigned to the project activity "Debugging"
in project ID
220001
When "alew" registers start time as "15-03-2022 12.00"
And registers end time as "15-03-2022 15.35" to project activity "Debugging"
Then the total time is rounded to 3.5 hours
And 3.5 hours is added to total time spent on project activity "Debugging"

Scenario: Register time for activity on a given date for a developer not on the activity.
Given that a "customer" project "NEM-ID" with start week "2022-04" and end week "2022-45" exists
And the project activity "Debugging" with start week "2022-05" and end week "2022-07" and time budget 20
exists
And that the developer "alew" is not assigned to the project activity "Debugging" in project ID 220001
When "alew" registers start time as "15-03-2022 12.00" and end time as "15-03-2022 15.35"
to project activity "Debugging"
Then return error message "Developer is not on the activity"

Scenario: Register time for activity on a given date with invalid developer initials
Given that a "customer" project "NEM-ID" with start week "2022-04" and end week "2022-45" exists
And the project activity "Debugging" with start week "2022-05" and end week "2022-07" and time budget 20
exists
And that the developer "alews" is assigned to the project activity "Debugging" in project ID
220001
Then return error message "Invalid amount of initials for developer name.
Initials length must be between 1 and 4 (inclusive)"

Scenario: Register time for activity on a given date with invalid developer initials
Given that a "customer" project "NEM-ID" with start week "2022-04" and end week "2022-45" exists
```

```
And the project activity "Debugging" with start week "2022-05" and end week "2022-07"
and time budget 20 exists
And that the developer "a.ex" is assigned to the project activity "Debugging" in project ID 220001
Then return error message "Illegal character(s) found. Only letters are allowed in developer
initials"

Scenario: Register time for activity on a given date and edit time afterwards
Given that a "customer" project "NEM-ID" with start week "2022-04" and end week "2022-45" exists
And the project activity "Debugging" with start week "2022-05" and end week "2022-07" and time budget 20
exists
And that the developer "alew" is assigned to the project activity "Debugging" in project ID 220001
When "alew" registers start time as "15-03-2022 12.00" and end time as "15-03-2022 15.35"
to project activity "Debugging"
Then the total time is rounded to 3.5 hours
And 3.5 hours is added to total time spent on project activity "Debugging"
When "alew" wants to edit registered time on project activity "Debugging"
Then the previous time registration for project activity "Debugging" of 3.5 hours with start time
"15-03-2022 12.00" and end time "15-03-2022 15.35" is deleted in project ID 220001
Then "alew" registers start time as "15-03-2022 13.00" and end time as "15-03-2022 15.35"
to project activity "Debugging"
Then the total time is rounded to 2.5 hours
```

```
Feature: Change end week for a project
  Description: Change end week for either a customer project or an internal project
  Actors: Project manager or Developer

  Scenario: Change end week for a customer project
    Given that the project "NEM-ID", for a "customer", with start week "2022-05"
    and end week "2022-50" exists
    When developer "asbg" changes end week to "2022-09" from "2022-08" for project "NEM-ID"
    Then end week for project "NEM-ID" is "2022-09"

  Scenario: Change end week for a non-existent customer project
    Given that the project "Togbillet", for a "customer", with start week "2022-05" and end week
    "2022-45" exists
    But that the developer "whkp" changes end week to "2022-45" from "2022-04" for project "Rejsekort"
    Then the errormessage "Project does not exist" is thrown

  Scenario: Enter the desired end week in a wrong format
    Given that the project "Firewall", for a "customer", with start week "2022-05" and end week
    "2022-45" exists
    But that the developer "alew" by accident changes end week to "2022-440", instead of "2022-40",
    from "2022-45" for project "Firewall"
    Then an error with errormessage "Wrong date format" is thrown


Feature: Change end week for an activity
  Description: Change end week for either a project activity or a personal activity.
  Actors: Project manager or Developer

  Scenario: Change end week for a external activity
    Given developer "asbg" exists
    And that external activity "Vacation to New York" with start week "2022-04" and end week "2022-10"
    with time budget 0 exists
    When developer "asbg" changes end week to "2022-09" for activity "Vacation to New York"
    Then end week for "Vaction to New York" is "2022-09"

  Scenario: Change end week for a external activity that does not exist
    Given developer "asbg" exists
```

```
    When developer "asbg" changes end week to "2022-09" for activity "whkp sick leave"
    Then the error message "External activity does not exist" is returned

  Scenario: Change end week for a project activity to after the project end week
    Given that project activity "Programming UI" exists in the "customer" project "NEM-ID" with
    start week "2022-09" and end week "2022-41"
    And that the project activity "Programming UI" has start week "2022-13", end week "2022-19"
    and time budget 20
    When project manager "asbg" changes end week to "2022-45"
    Then the end week of project "NEM-ID" is changed to match the new end week of activity
    "Programming UI"

Feature: Create project activity
  Description: Add a project activity to an existing project.
  Actors: Project manager

  Scenario: Add project activity "UX-Design" to the project "NEM-ID" as the project manager
  of the project "NEM-ID"
    Given that the "customer" project named "NEM-ID" with start week "2022-09"
    and end week "2022-50" exists
    And that the user "whkp" is the project manager of the project "NEM-ID"
    Then the user "whkp" creates a new project activity "UX-Design", start week "2022-14",
    end week "2022-16" and time budget 20 hours for project "NEM-ID"
    Then the new project activity "UX-Design" with start week "2022-14", end week "2022-16"
    and time budget 20 hours for project "NEM-ID" is created

  Scenario: Add project activity "UX-Design" to the project "MIT-ID" when the user is not
  the project manager of "MIT-ID"
    Given that the "customer" project named "NEM-ID" with start week "2022-09" and
    end week "2022-50" exists
    And that the user "whkp" is not the project manager of the project "NEM-ID"
    Then the user will receive error message "You are not project manager"

Feature: Assign developer to project
  Description: A feature that allows a project manager of a project to assign a
  developer to one of their projects
  Actors: Project Manager

  Scenario: Assign developer to a customer project
    Given that the developer "whkp" is the project manger of the "customer" project "NEM-ID"
    with start week "2022-19" and end week "2022-43"
    And developer "dprk" exists
    And developer "dprk" is not already a developer on the "customer" project "NEM-ID"
    Then project manager "whkp" adds developer "dprk" to "customer" project "NEM-ID"

  Scenario: Assign developer to a customer project when they have already been added to the project
    Given that the developer "whkp" is the project manger of the "customer" project "NEM-ID"
    with start week "2022-19" and end week "2022-43"
    And developer "dprk" exists
    And developer "dprk" is already a developer on the "customer" project "NEM-ID"
    Then return the error message "This developer has already been assigned to the project"


Feature: Assign developer to project activity
  Description: A feature that allows a project manager of a project to assign a developer
  to a project activity in one of their projects.
  Actors: Project Manager

  Scenario: Assign developer to a customer activity
    Given that developer "whkp" exists
    And "whkp" is the project manger of the "customer" project "NEM-ID" with start week
    "2022-32" and end week "2023-09"
```

```
    And the project activity "UX-Design" with start week "2022-52", end week "2023-08"
    and time budget 20 exists for the project.
    Given that "dprk" is assigned to the project "NEM-ID"
    And that "dprk" is a free employee in the period from "2022-52" to "2023-08"
    Then developer "dprk" will be assigned to the project activity "UX-Design"

Scenario: Attempt to assign developer to a customer activity that doesn't exist
    Given that developer "asbg" exists
    And "asbg" is the project manger of the "internal" project "Rejsekort"
    with start week "2022-03" and end week "2023-02"
    Given that "whkp" is assigned to the project "Rejsekort"
    And that "whkp" is a free employee in the period from "2022-40" to "2023-45"
    When developer "whkp" is assigned to the project activity "Payment transfer" that does not exist
    Then the errormessage "Activity \"Payment transfer\" could not be found"

Feature: Create Report
  Description: A weekly text containing a summary of a projects' status
  Actors: Project manager or developer

    Scenario: Create a report for the customer project "DTU Website"
        Given that the developer "whkp" and "customer" project "DTU Website" with
        start week "2022-02", end week "2022-32" exists
        And that "whkp" is the project manager of the project "DTU Website"
        And the project "DTU Website" contains the first activity "Create UI" with
        start week "2022-04" and end week "2022-07" and time budget 50
        And the project "DTU Website" contains the second activity "Optimise server"
        with start week "2022-25" and end week "2022-30" and time budget 80
        And "whkp" creates a weekly report for project "DTU Website"
        Then a report for "DTU Website" is created containing name of the project,
        project type, timebudget, total time spent on the project and
        the estimated work time that is left on the project

    Scenario: Create a report for the customer project "DTU Website"
    if you are not a project manager
        Given that the developer "nikh" and "customer" project "DTU Website"
        with start week "2022-02", end week "2022-32" exists
        And that "nikh" is not the project manager of the customer project "DTU Website"
        And "nikh" creates a weekly report for project "DTU Website"
        Then it returns the error message "You are not project manager"

    Scenario: Create a report for the internal project "Fix SOFTWAREHUSET's website"
        Given that the developer "asbg" and "internal" project "Fix SOFTWAREHUSET's website"
        with start week "2022-01", end week "2022-03" exists
        And that "asbg" is the project manager of the project "Fix SOFTWAREHUSET's website"
        And "asbg" creates a weekly report for project "Fix SOFTWAREHUSET's website"
        Then a report for "Fix SOFTWAREHUSET's website" is created containing
        name of the project, project type, time budget, total time spent on the project
        and the estimated work time that is left on the project
```

3.4 Appendix D: Class Diagram

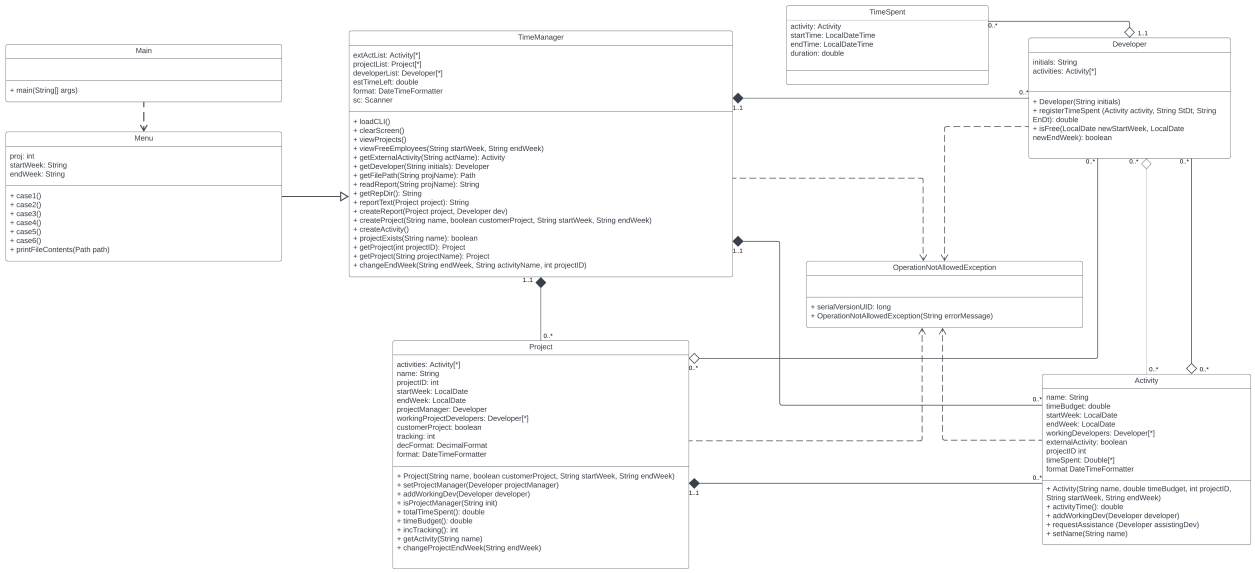


Figure 2: Class Diagram

3.5 Appendix E: Sequence Diagram

Feature 1, Create project

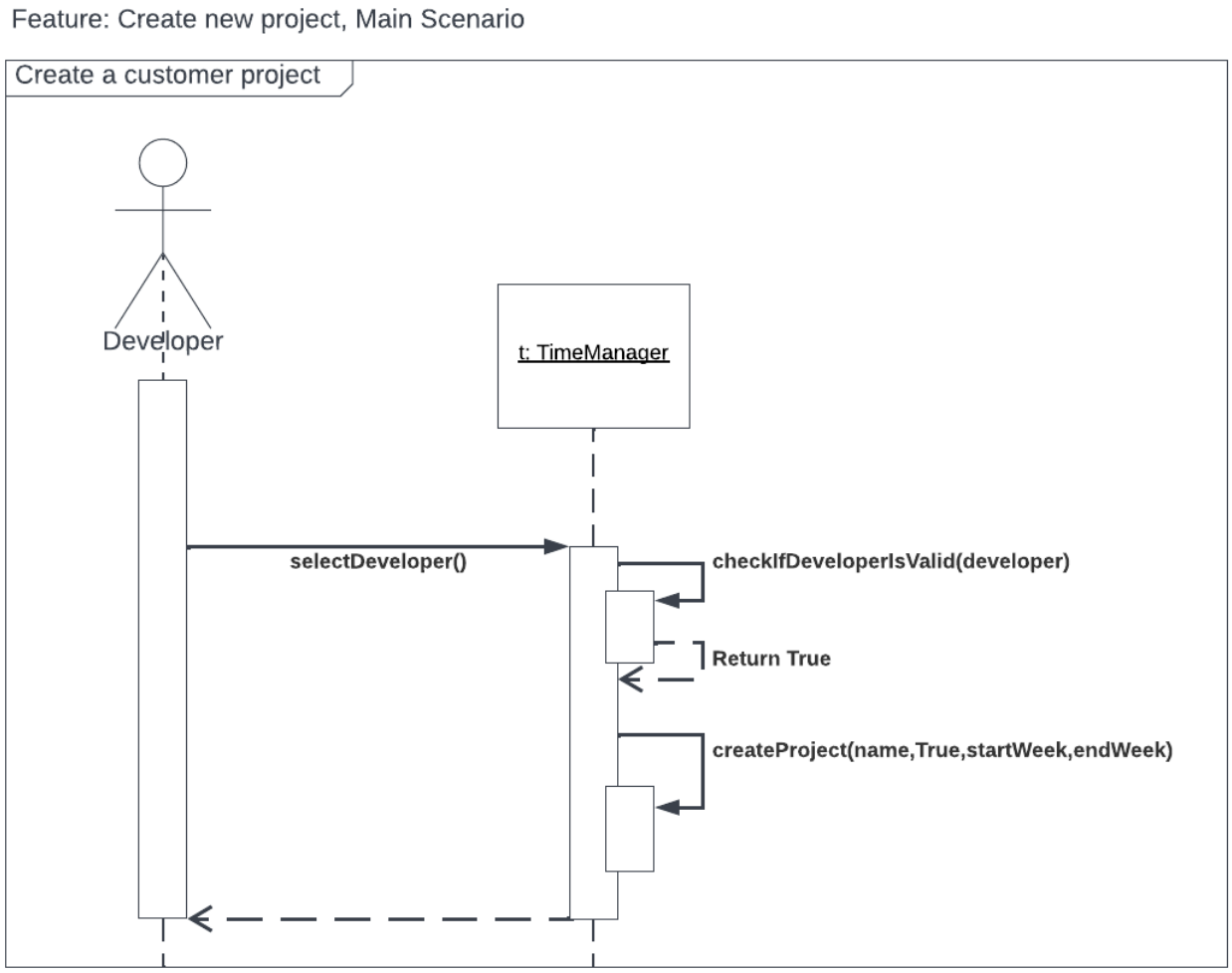


Figure 3: Create a customer project

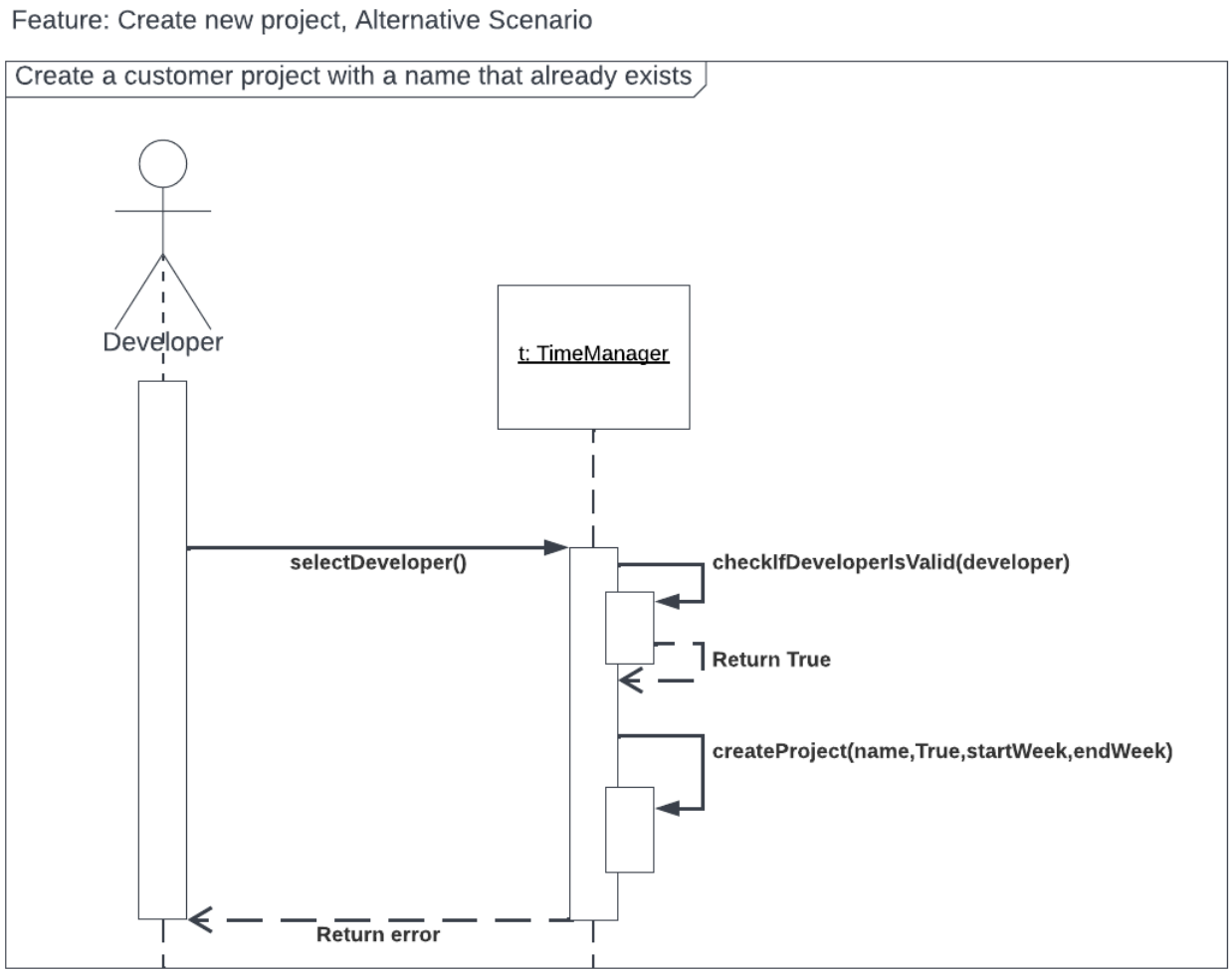


Figure 4: Create a customer project with a name that already exists

Feature 2, Create external activity

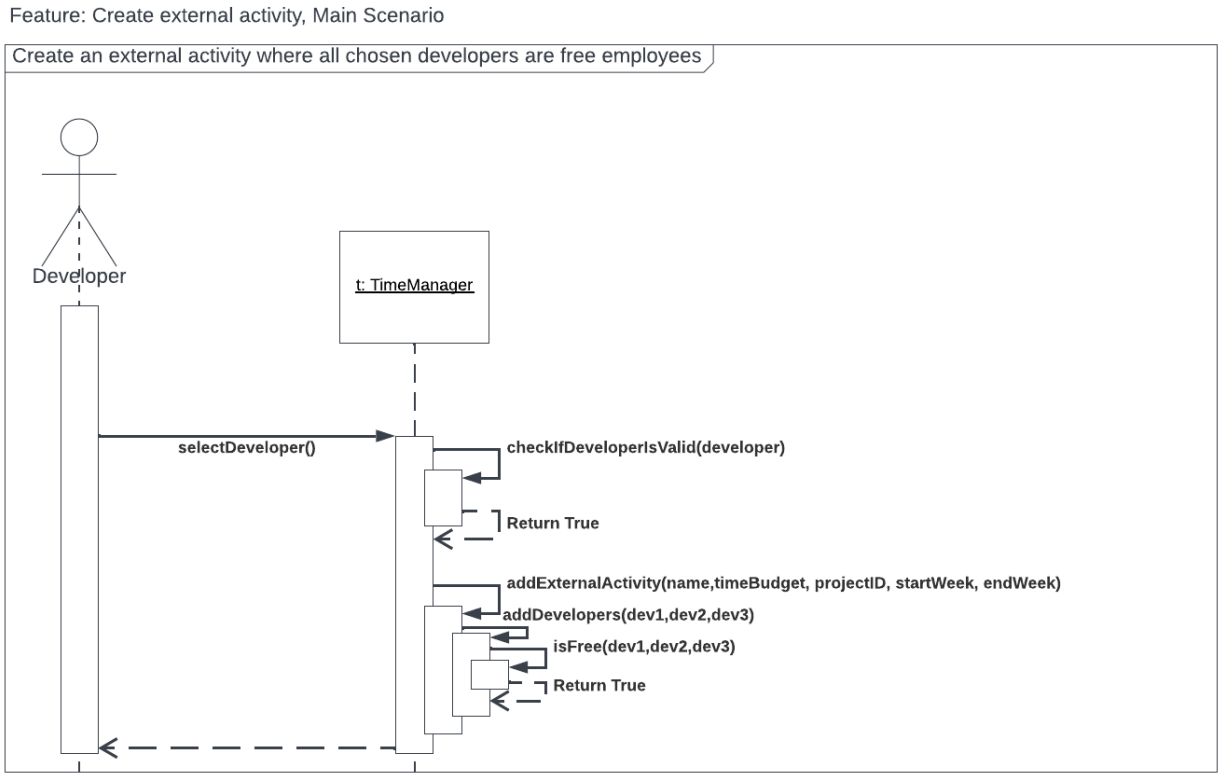


Figure 5: Create an external activity where all chosen developer are free employees

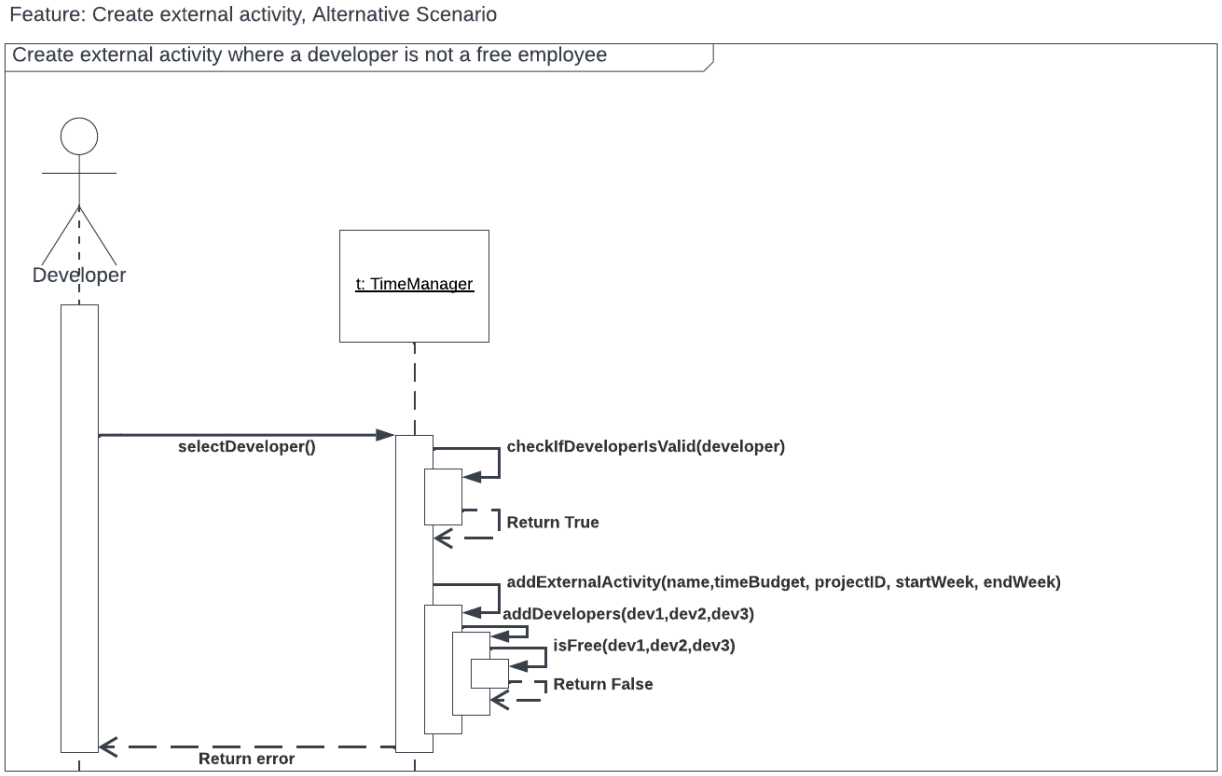


Figure 6: Create external activity where a developer is not a free employee

Feature 3, Assign a project manager to a project

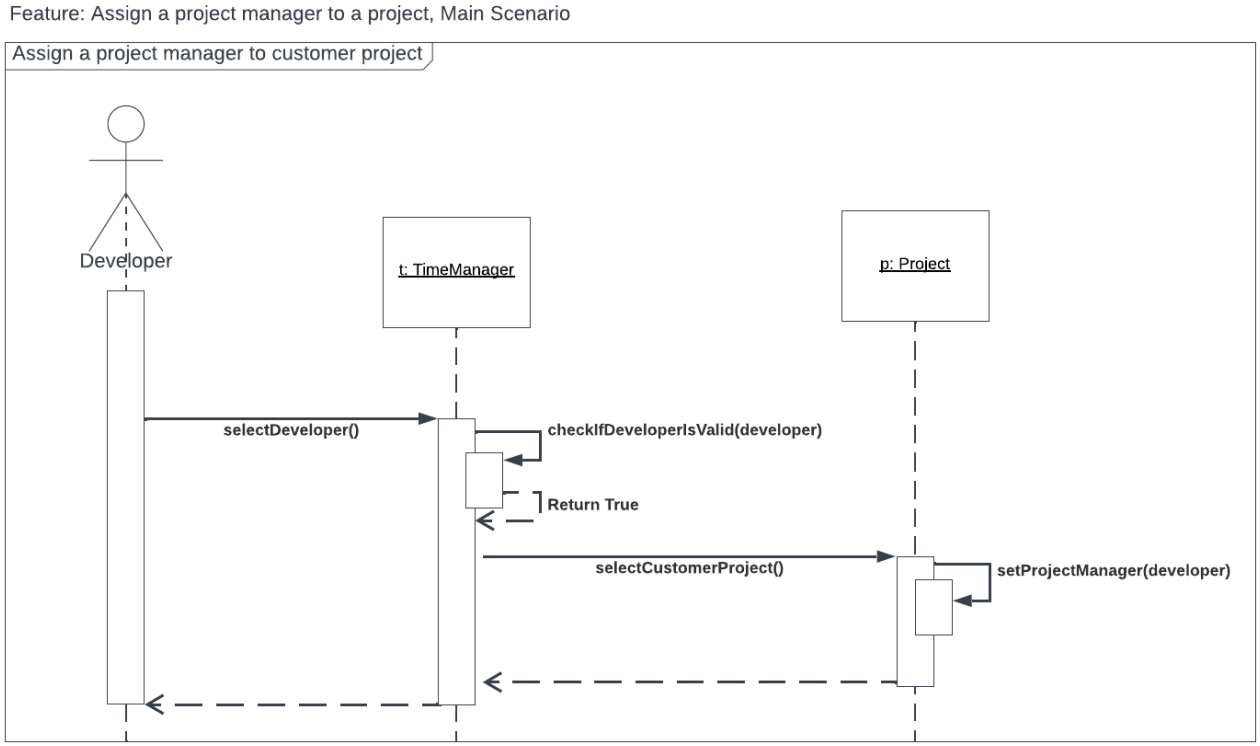


Figure 7: Assign a project manager to a customer project

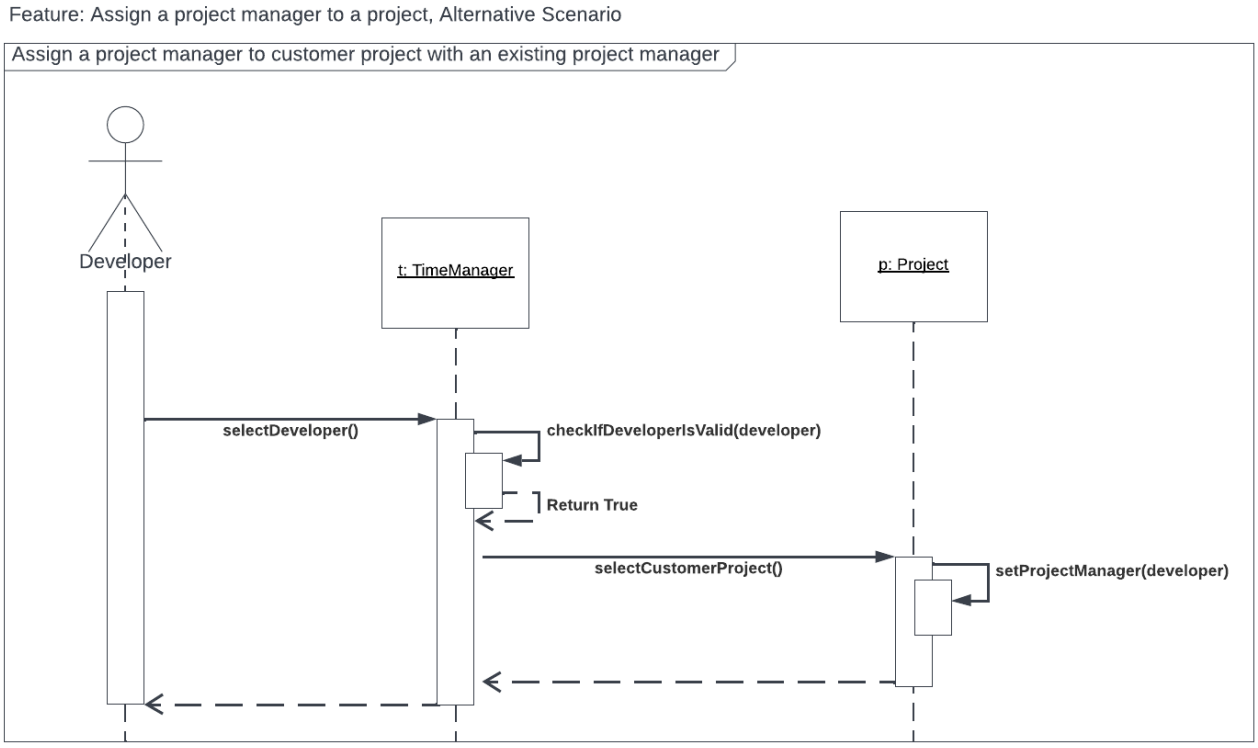


Figure 8: Assign a project manager to a customer project with an existing project manager

Feature 4, Change end week for project

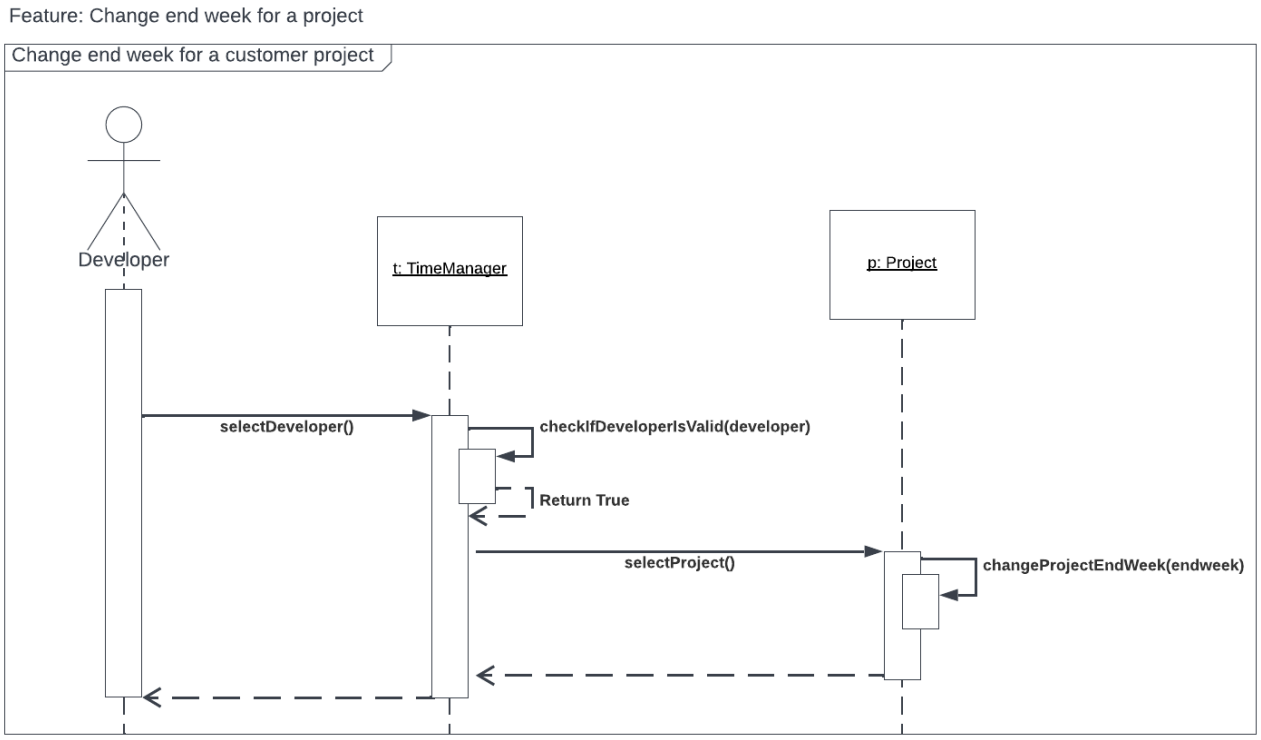


Figure 9: Change end week for a customer project

Feature 5, Change end week for an activity

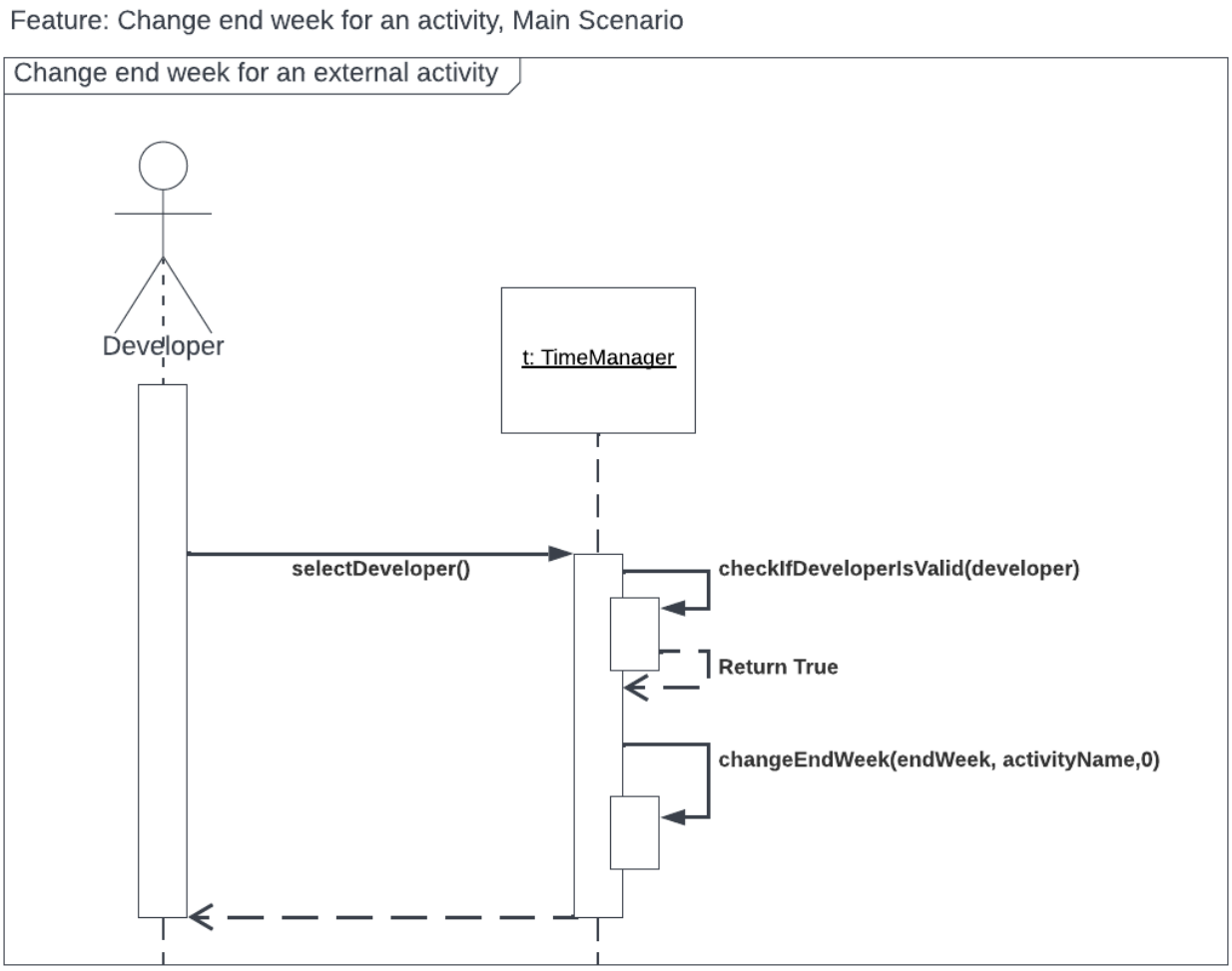


Figure 10: Change end week for an external activity

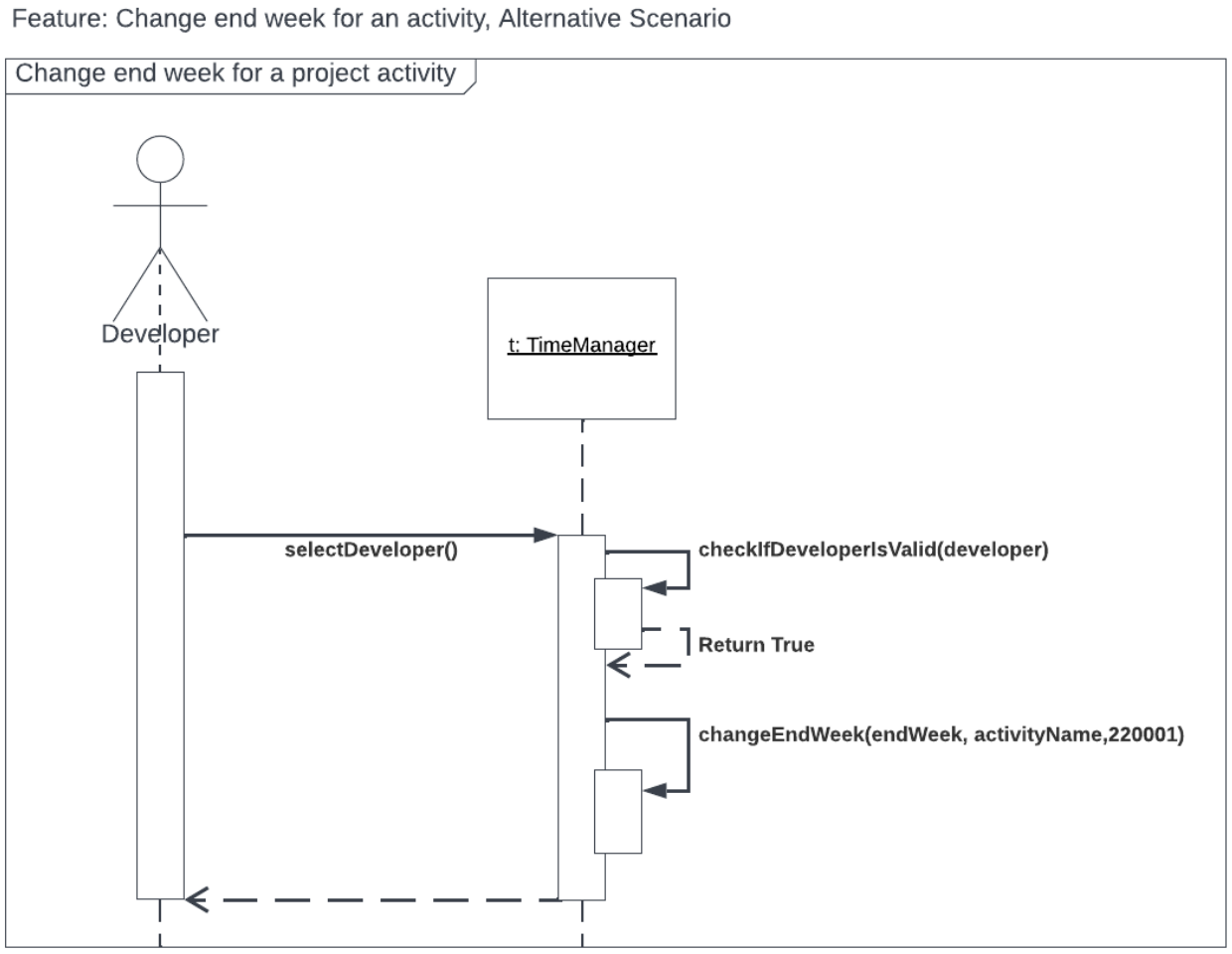


Figure 11: Change end week for a project activity

Feature 6, Create Project Activity

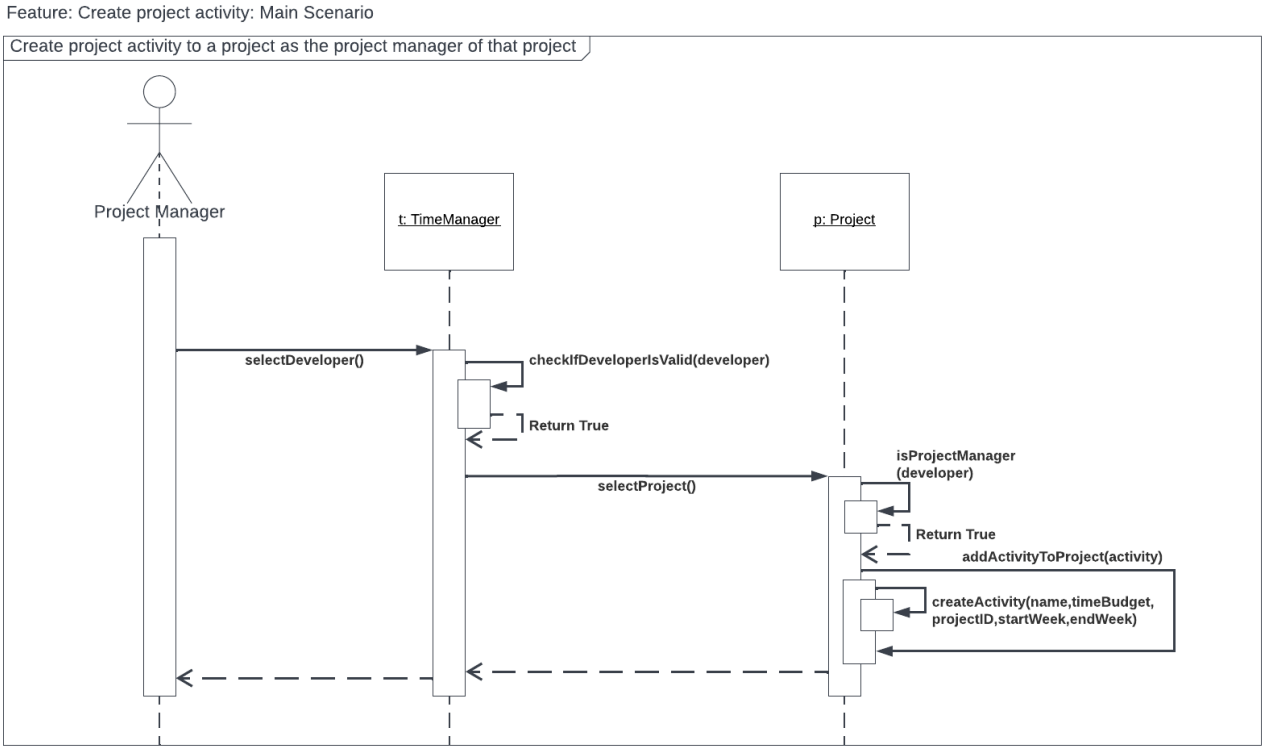


Figure 12: Create project activity to a project as the project manager of that project

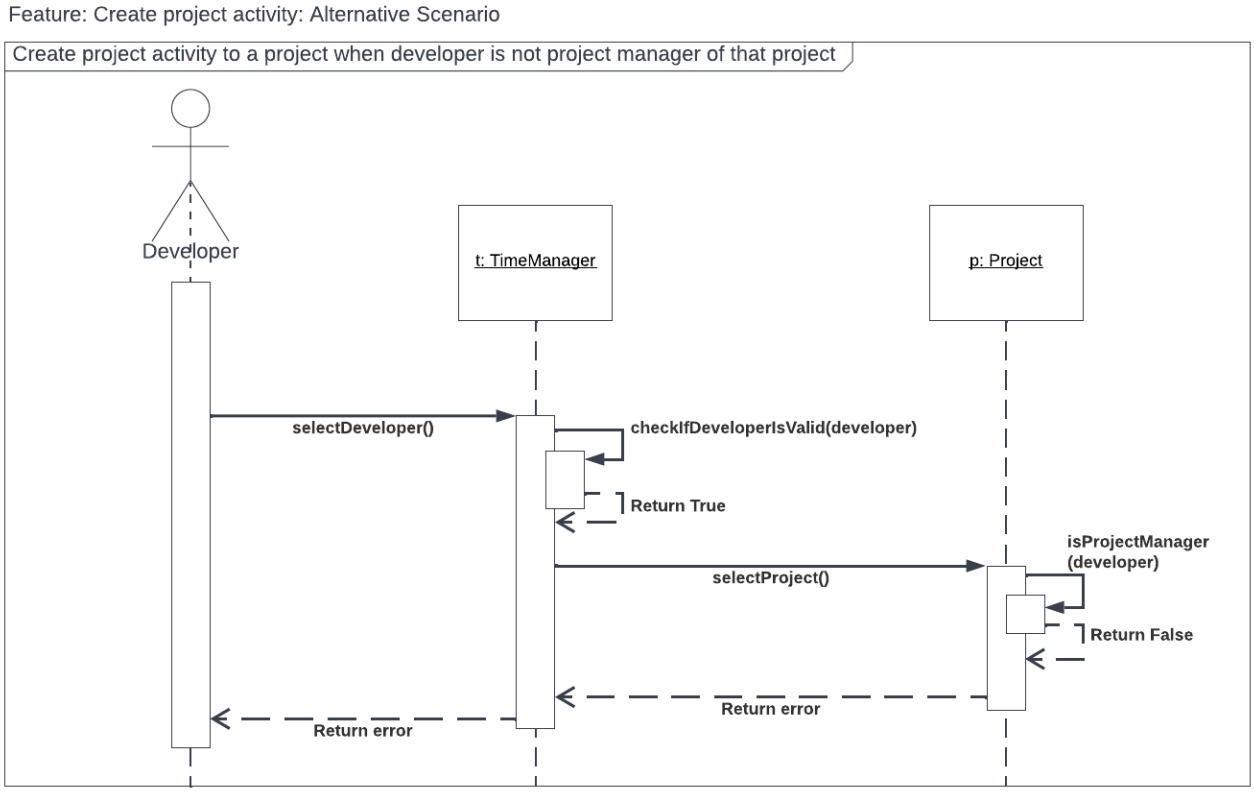


Figure 13: Create project activity to a project when developer is not project manager of that project

Feature 7, Assign developer to project

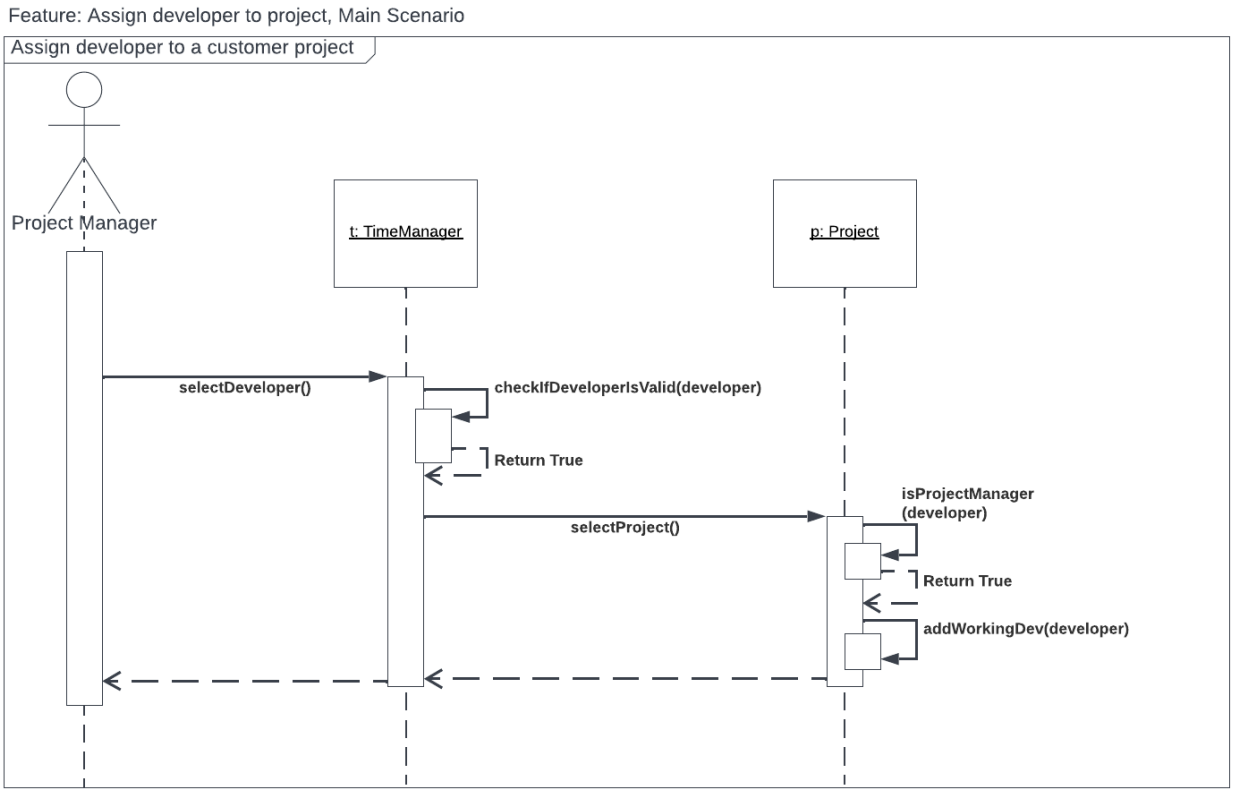


Figure 14: Assign developer to a customer project

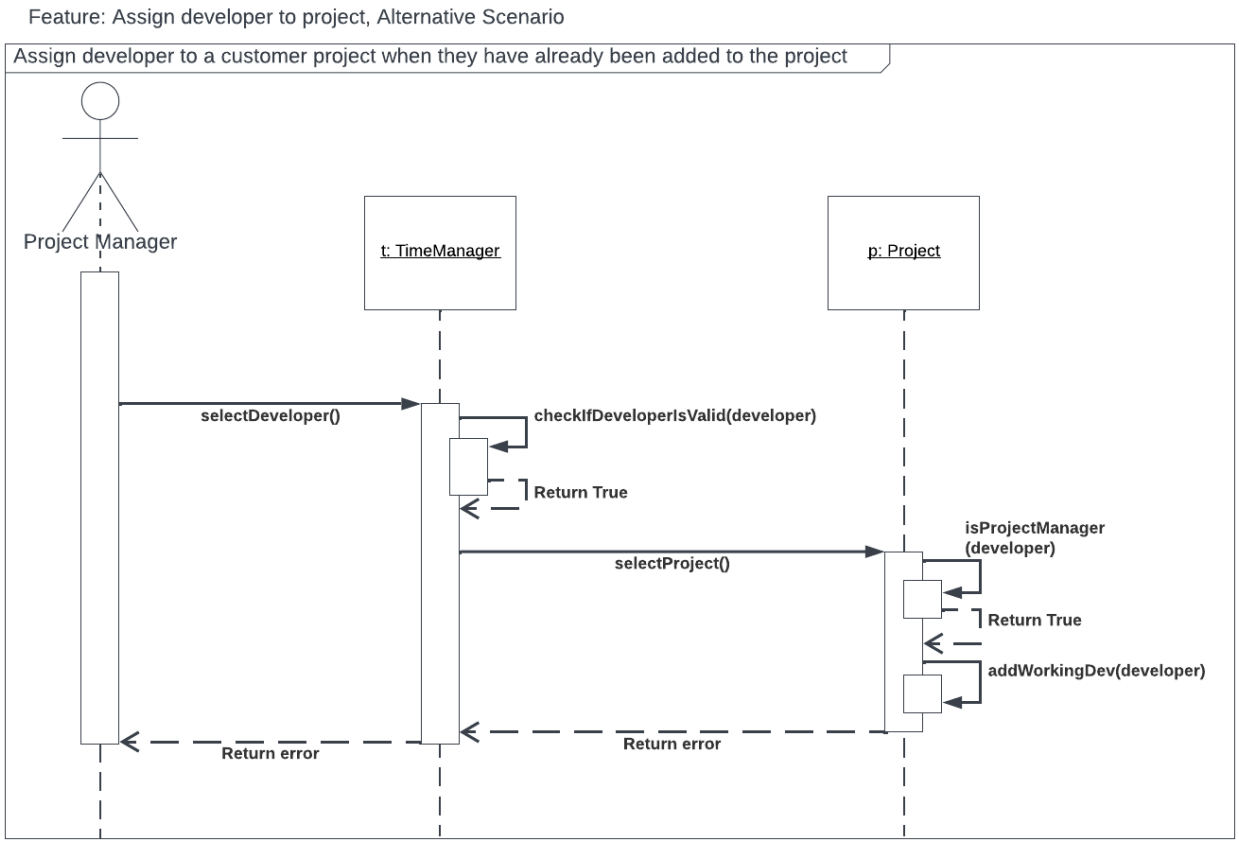


Figure 15: Assign developer to a customer project when they have already been added to the project

Feature 8, Assign developer to project activity

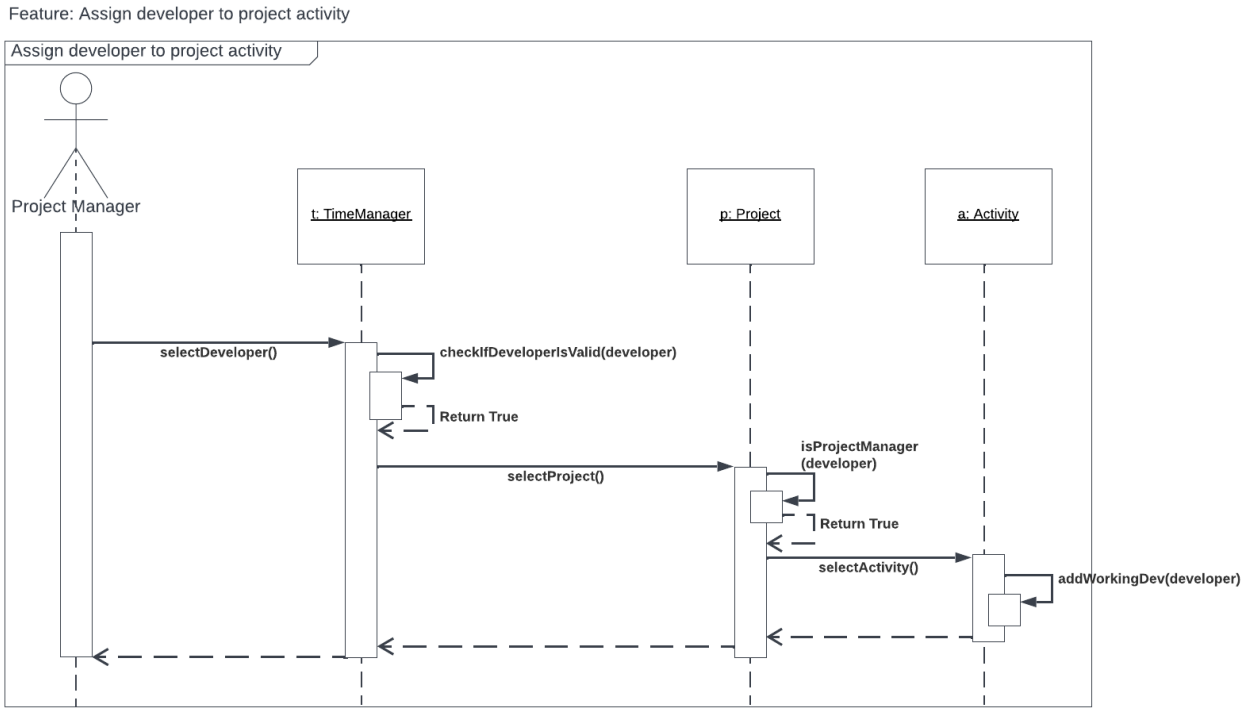


Figure 16: Assign developer to project activity

Feature 9, Create Report

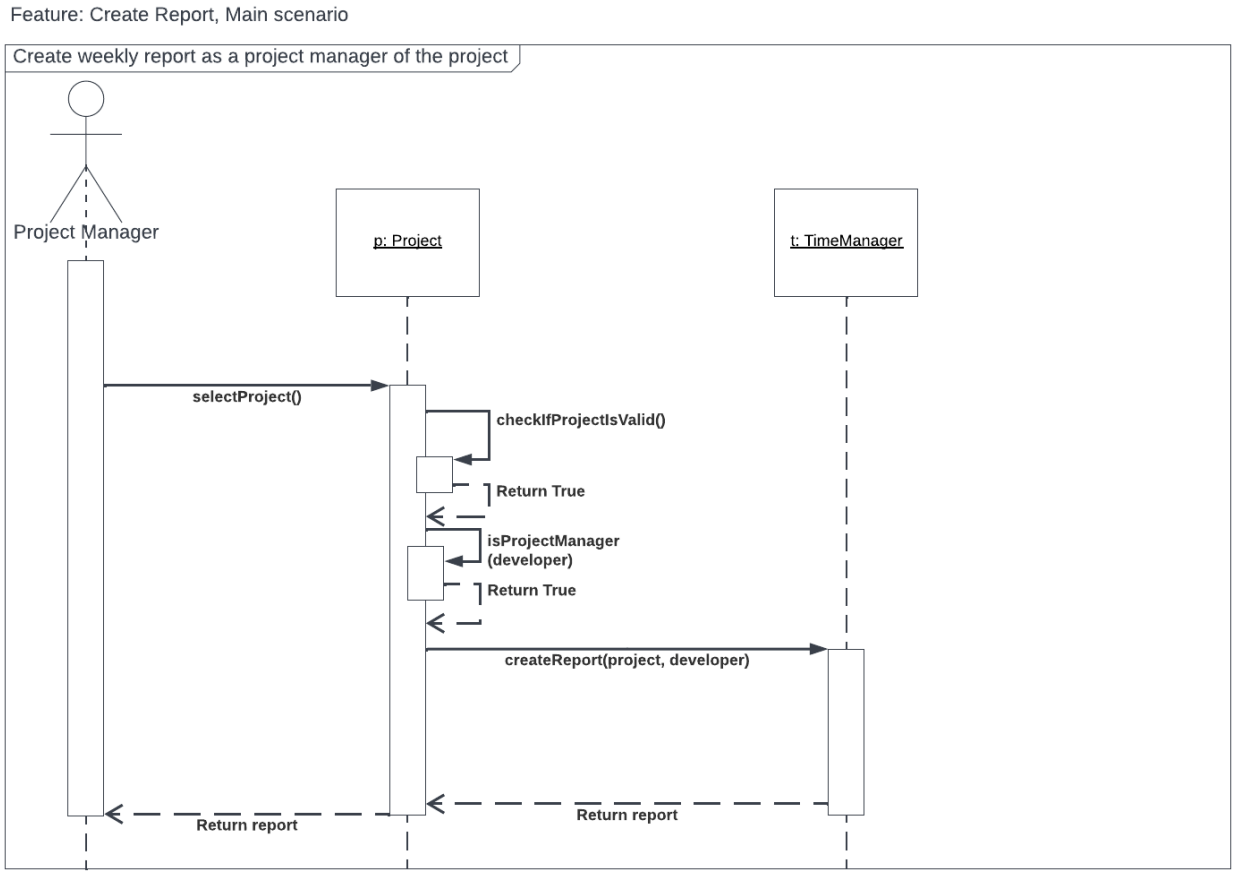


Figure 17: Create weekly report as a project manager of the project

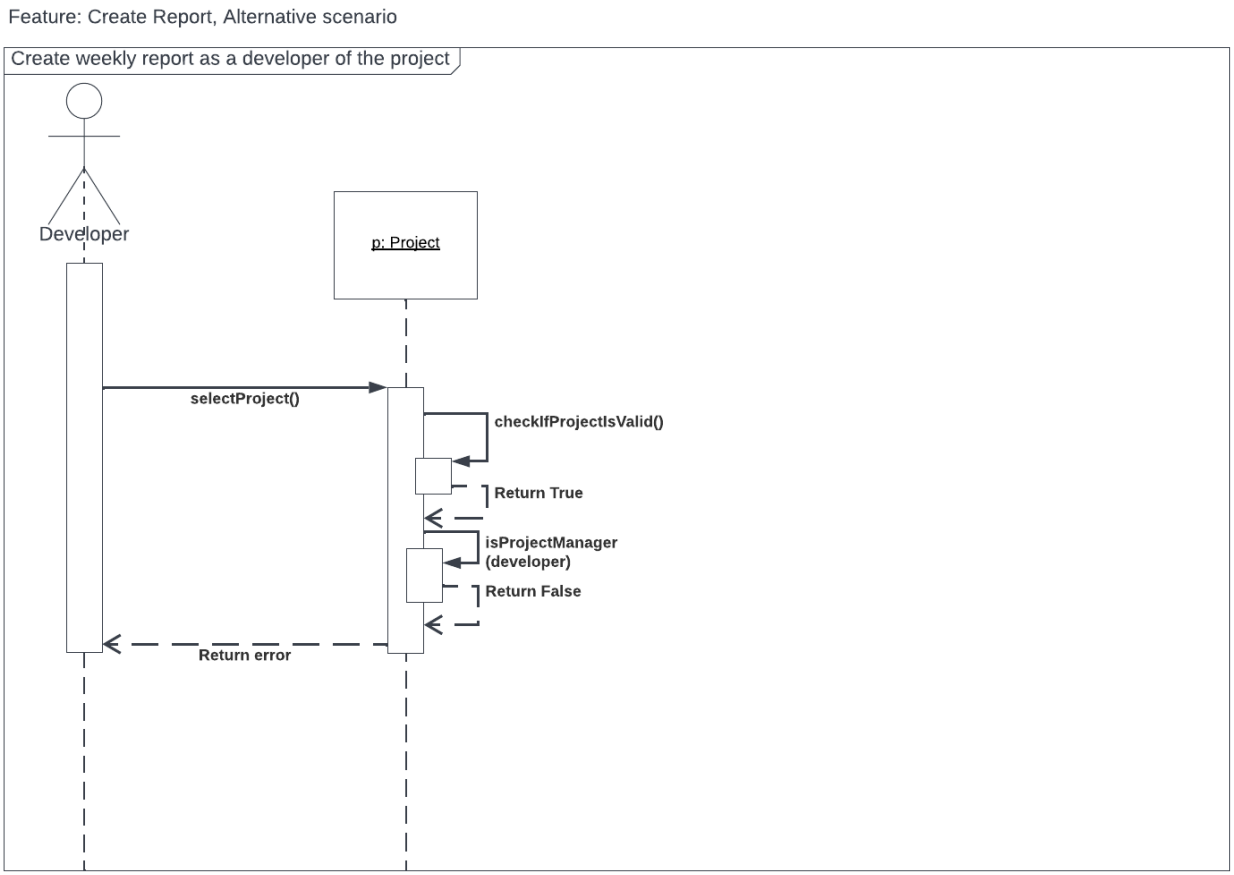


Figure 18: Create weekly report as a developer of the project

Danmarks Tekniske Universitet

Project Management System - Part 2

Group 01
Aleksander Wind (s214683)
Alexander Bendix Gosden (204209)
Nikolai Hansen (214681)
William Peytz (s204145)

Software Engineering, 02161

Technical University of Denmark
April, 2022

Contents

Work distribution

	Aleksander W.	Alexander G.	Nikolai	William
Systematic tests:	createProject	registerTimeSpent	createReport	isFree
Code coverage:	40%	10%	40%	10%
Design by contract:	createProject	registerTimeSpent	createReport	isFree
Design patterns/SOLID Principles:	10%	40%	10%	40%
Conclusion:	10%	40%	10%	40%

1 Systematic tests (white box tests)

White box testing is a software testing method software developers use to test their code, when the internal structure of the code is known. Testing primarily focuses on the testing of code structure, branches, loops, conditions etc. When doing a white box test the only focus is the method at hand and not the method that method is using. This is done under the assumption that there are seperate white box test for these methods, even though they won't necessarily be done as in our case. We have chosen to create systematic tests for four methods in our project: createProject, isFree, registerTimeSpent and createReport.

1.1 Method 1: createProject

```
public void createProject(String name, boolean customerProject, String startWeek, String endWeek) throws OperationNotAllowedException
{
    if (!projectExists(name)) // 1
    {
        projectList.add(new Project(name, customerProject, startWeek, endWeek)); // 2
    }
    else
    {
        throw new OperationNotAllowedException("Project could not be created, as the project name is already in use."); // 3
    }

}
```

Figure 1: Relevant code for method createProject. The first is the constructor for the Project class and the method createProject is from class TimeManager

Path	Input data set	Input property
1 true, 2	A	Project which already exists
1 false, 3	B	Project which does not already exist and end week is after start week
1 false, 3 (constructer throws exception)	C	Project which does not already exist and end week is before start week

Input data set	Contents	Expected Output
A	projectList contains project with name "NEM-ID"; name ="NEM-ID",customerProject=True, startWeek=2022-03,endWeek=2022-50	projectList remains unchanged. Returns error "Project could not be created, as the project name is already in use."
B	projectList does not contain project with name "NEM-ID"; name ="NEM-ID", customerProject=True, startWeek=2022-03,endWeek=2022-50	Project with name "NEM-ID" is added to projectList
C	projectList does not contain project with name "NEM-ID"; name ="NEM-ID",customerProject=True, startWeek=2022-50, endWeek=2022-25	projectList remains unchanged. Returns error "End week of project is before start week"

1.2 Method 2: isFree

```
public boolean isFree(LocalDate newStartWeek, LocalDate newEndWeek) throws OperationNotAllowedException
{
    for (LocalDate i = newStartWeek; i.isBefore(newEndWeek);i = i.plusWeeks(1))
    {
        int activityCount = 0;
        for (Activity a : activities)
        {
            if (activityCount >= 20) //1
            {
                throw new OperationNotAllowedException("Developer is not free in the given time period."); //2
            }
            else if ((i.isAfter(a.startWeek)) && (i.isBefore(a.endWeek))) //3
            {
                activityCount++; //4
            }
        }
    }
    return true; //5
}
```

Figure 2: Relevant code for method isFree, which is in the class Developer

Path	Input Data	Input Property
1 true, 2	A	Developer has 20 activities in the given time period and is therefore not free
1 false, 3 true, 4, 5	B	Developer has between 1 and 19 activities in the given time period and is therefore free
1 false, 3 false, 5	C	Developer has no activities in the given time period and is therefore free

Input data set	Contents	Expected Output
A	arrayList activities for developer "whkp" contains 20 activities with name = "Meeting0", "Meeting1", ..., "Meeting19" timeBudget=20, projectID=220001, startWeek=2022-05, endWeek=2022-25	Returns error "Developer is not free in the given time period.". since developer has 20 active activities in the time period newStartWeek=2022-07, newEndWeek=2022-17
B	arrayList activities for developer "whkp" contains 2 activities with name = "Meeting0", "Meeting1", timeBudget=20, projectID=220001, startWeek=2022-05, endWeek=2022-25	Returns true, since developer has 2 active activities in the time period newStartWeek=2022-07, newEndWeek=2022-17
C	arrayList activities for developer "whkp" contains 0 activities	Returns true, since developer has 0 active activities in the time period newStartWeek=2022-07, newEndWeek=2022-17

1.3 Method 3: registerTimeSpent

```
public double registerTimeSpent(Activity activity, String StDt, String EnDt) throws OperationNotAllowedException
{
    LocalDateTime startTime;
    LocalDateTime endTime;
    double prevActTime = activity.activityTime();
    if (!activity.workingDevelopers.contains(this)) //1
    {
        throw new OperationNotAllowedException("Developer is not on the activity"); //2
    }
    DateTimeFormatter format = DateTimeFormatter.ofPattern("dd-MM-yyyy HH.mm");
    try
    {
        startTime = LocalDateTime.parse(StDt, format); //3
        endTime = LocalDateTime.parse(EnDt, format); //4
    }
    catch (DateTimeParseException ONAE)
    {
        throw new OperationNotAllowedException("The entered dates are not of the right format."); //5
    }

    double l = Duration.between(startTime, endTime).toMinutes(); //6
    if (l <0) //7
    {
        throw new OperationNotAllowedException("Start time is after end time"); //8
    }
    l = (double) Math.round(l/30)/2; //9
    activity.timeSpent.add(l); //10
    workTimes.add(new TimeSpent(activity, startTime, endTime, l)); //11
    return l; //12
}
```

Figure 3: Relevant code for method registerTimeSpent, which is in class Developer

Path	Input data set	Input property
1 true, 2	A	Developer is not on the activity
1 false, 3, 4, 6, 7 false, 9, 10, 11, 12	B	Time is registered to the activity
1 false, 3, 4, 6, 7 true, 8	C	Start time is after end time
1 false, 3, 5	D	Format of start time is wrong
1 false, 4. 5	E	Format of end time is wrong

Input data set	Contents	Expected Output
A	Developer "alew" is not assigned to the project activity "Debugging", in customer project "NEM-ID", project ID 22001. Registers start time as "15-03-2022 12.00" and end time as "15-03-2022 15.35" to project activity "Debugging"	Returns error message: "Developer is not on the activity"
B	Developer "alew" is assigned to the project activity "Debugging" in customer project "NEM-ID", project ID 22001 Registers start time as "15-03-2022 12.00" and end time as "15-03-2022 15.35" to project activity "Debugging"	3.5 hours is added to the total time spent on project activity "Debugging"
C	Developer "alew" is assigned to the project activity "Debugging" in customer project "NEM-ID", project ID 22001. Registers start time as "15-03-2022 15.35" and end time as "15-03-2022 12.00" to project activity "Debugging"	Returns error message: "Start time is after end time"
D	Developer "alew" is assigned to the project activity "Debugging" in customer project "NEM-ID", project ID 22001. Registers start time as "15-03-22 12" and end time as "15-03-2022 15.35" to project activity "Debugging"	Returns error message: "The entered dates are not of the right format"
E	Developer "alew" is assigned to the project activity "Debugging" in customer project "NEM-ID", project ID 22001. Registers start time as "15-03-22 12" and end time as "15-03-2022 12.00" and end time as "15-03-22 15" to project activity "Debugging"	Returns error message: "The entered dates are not of the right format"

1.4 Method 4: createReport

```
public void createReport(Project project, Developer dev) throws OperationNotAllowedException, IOException
{
    project.isProjectManager(dev.initials); //1
    String report = reportText(project); //2
    String projRepDir = getRepDir(); //3
    File projReport = new File( pathname: projRepDir + project.projectID + "_report.txt"); //4
    projReport.createNewFile(); //5
    FileWriter FlWrtr = new FileWriter(projReport); //6
    FlWrtr.write(report); //7
    FlWrtr.close(); //8
    System.out.println("Project report has been saved in: " + getRepDir());
}
```

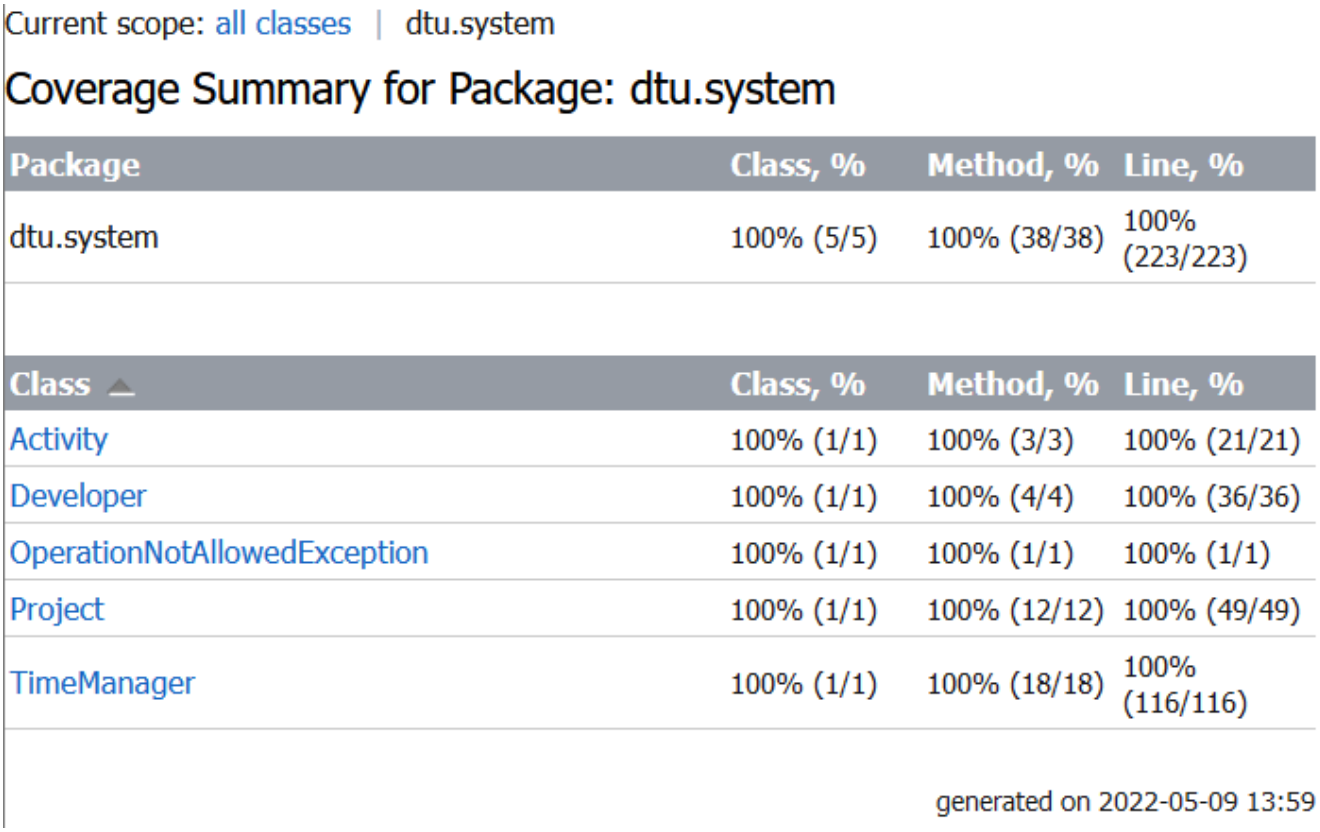
Figure 4: Relevant code for method createReport, which is in class TimeManager

Path	**Input data set**	**Input property**
1 true, 2, 3, 4, 5, 6, 7, 8	A	Developer is the project manager for the project
1 false	B	Developer is not project manager for the project
1 false	C	Project does not exist

Input data set	Contents	Expected Output
A	Developer "whkp" is project manager for the customer project "DTU website" with startWeek "2022-02", endWeek "2022-32".	A project report for "DTU website" is created as a .txt file and the message "Project report has been saved in C:\\Users\\whkp\\Documents\\ProjectReports\\" is received.
B	Developer "nikh" is not project manager for the customer project "DTU website" with startWeek "2022-02", endWeek "2022-32".	Returns error message "You are not project manager"
C	Developer "asbg" tries to create report for non-existing project "Fix SOFTWAREHUSET's website"	Returns error message "Project does not exist"

2 Code coverage

Code coverage is a tool to measure how much of the source code is executed when run through tests. This is important to tell if there are any functionalities in the program that possibly haven't been tested throuroughly yet. When running the Cucumber tests with Code Coverage, we get the following distribution:



Current scope: all classes | dtu.system

Coverage Summary for Package: dtu.system

Package	Class, %	Method, %	Line, %
dtu.system	100% (5/5)	100% (38/38)	100% (223/223)

Class	Class, %	Method, %	Line, %
Activity	100% (1/1)	100% (3/3)	100% (21/21)
Developer	100% (1/1)	100% (4/4)	100% (36/36)
OperationNotAllowedException	100% (1/1)	100% (1/1)	100% (1/1)
Project	100% (1/1)	100% (12/12)	100% (49/49)
TimeManager	100% (1/1)	100% (18/18)	100% (116/116)

generated on 2022-05-09 13:59

Figure 5: Code coverage

The Code Coverage data consists of the classes `Activity`, `Developer`, `OperationNotAllowedException`, `Project` and `TimeManager`. As seen there is 100% code coverage, meaning every functionality of our program has been tested. The classes `Menu` and `Main` are used solely for the UI, hence the reason they are not included in the Cucumber tests, and by extension, the test for Code Coverage. The `TimeSpent` class is also not included in the test for Code Coverage, as it is just a collection of objects and has no real functionality beyond being a container of different objects.

3 Design by contract

We have chosen to express the logic through code in our design by contracts instead of using discrete mathematical notation, as we feel that this expresses our conditions better. The pre- & post-conditions have been implemented in their respective methods with assert statements. All of our pre-conditions have been set to true, such that the program is more client friendly.

In the first method for createProject we set the post-condition to either return that the end week is before the start week of the project, or return that the project has been created.

In the second method for isFree we set the post-condition to either return that the developer in the given time period has less than 20 active activities (i.e. that isFree returns true and therefore the developer is free) or returns the error message "Developer is not free in the given time period", i.e. they have ≥ 20 activities.

In the third method for registerTimeSpent we set the post-condition to return that the current activity time minus the time added to the previous activity time l, is equal to the previous activity time.

In the fourth method for createReport we set the post-condition to return that a report for the given projectID has been created.

3.1 Method 1: createProject

Contract for createProject:

pre: true

post: !toLocalDate(startWeek).isBefore(toLocalDate(endWeek)) == true ||
projectExists(projectName) == true

3.2 Method 2: isFree

Contract for isFree:

pre: true
post: activityCount<20 || "Developer is not free in the given time period"

3.3 Method 3: registerTimeSpent

Contracts for registerTimeSpent

pre: true

post: activity.activityTime()-l == prevActTime

3.4 Method 4: createReport

Contracts for createReport

pre: true
post: getReportFromProjectID != null

4 Design patterns/SOLID principles

4.1 Design Patterns

The TimeManager software uses different design patterns used in object oriented programming. It uses both creational and structual design patterns. In the following section are a few examples of some of the different design patterns from the software:

Creational patterns in the TimeManager software:
The singleton design pattern is a pattern that restricts the amount of instances of a class to a single instance. The singleton design pattern is used as there should only be one instance of `TimeManager/Menu`, as it is the central controlling class of the program. It handles the main functions of the program and interacts with all other classes.

Structural patterns in the TimeManager software:
We use the facade design pattern to mask the complexity of the `Project, Activity` and `Developer` classes. This is done with the `TimeManager` class, which also functions as an interface for the end user with the `Menu` subclass.

Other design patterns may also have been used, but it has not been a main focus of the developement.

4.2 SOLID principles

The TimeManager software follows the single-responsibility principle as all classes have been designed so they each only have one responsibility. For example the class `Project` is only responsible for modifying, getting, setting and creating/adding things related to objects of type `Project`. One method for modifying in class `Project` is `changeProjectEndWeek(String endWeek)`, which modifies a given project's end week. One method for getting is `getActivity(String name)`, which gets an activity from the correlating project, if there is one. Since the `getActivity(String name)` is related to the class `Project` and gets an activity, which is stored in an ArrayList in class `Project`, this method upholds the single-responsibility principle from our point of view. One method for setting is `setProjectManager(Developer projectManager)`, which assigns a project manager to the given project and therefore gives that developer privileges, to create reports and other things related to the project, that normal developers can't do. One method for creating/adding is `addWorkingDev(Developer developer)`, which adds a developer to the arraylist of workingProjectDevelopers.

The TimeManager software also follows the open-closed principles which means that all software entities have the ability to be extended without the need to modify the existing source code. For example the class `Activity` can easily be extended to have a class `Meetings`, in which meetings in activities are collected into this class. However this is one of many examples.

However the TimeManager software does not use all of the different SOLID principles. The Liskov substitution principle, has not been used as we didn't find it nessercary for the scope of this project. This also applies to the Interface segregation principle and the dependence inversion principle. Some of these principles could have been used if we decided to do make a graphical user interface, but we decided to make a command line interface instead. In general some of the SOLID-principles have been used more than others in the project.

5 Conclusion

5.1 Assesment of the final product

When working on a project of this size, within a limited time-period one has to make decisions in regard to scope and focus of the project, as there simply is not enough time to do everything. For our project we decided that it was important to have a functioning program which satisfied many of the clients requirements, while at the same time making sure we had 100% code coverage and extensive cucumber tests. This meant that we decided to focus less on subjects such as Design Patterns and SOLID principles, as they were introduced after the main part of our code was already made. However when writing the code we still had in mind making good and readable quality code.
Some features such as a login system or a GUI were not specific client requirements, and we therefore decided to leave them out. Overall we set high ambitions for the program and our final product satisfied these goals in the end.

5.2 Reflections on the workflow of the project

For the final part of the report we want to reflect on the workflow of the project as a whole. Since creating the original report 1, we found out we had to make a lot of adjustments in order to better reflect the final project. We learned a lot from the feedback on our report, as well as looking at another groups example. From this we were able to fix a lot of consistency issues, as well as streamline a lot of the report. We also simplified several of the diagrams, as we learned more about what was actually nessercary and what was required, in order to implement the project. We also made several changes to report 1, while actually developing the software, as we got a better scope of what methods were neeeded to complete the assignment. Overall report 1 ended up very closely resembling our final project.
In terms of how the work was done and distributed, we tried to work continously on the product every week, in order to get more time to think about and work on the problems. This was generally a good idea, but it also had some drawbacks, as we had not been presented with all of the course material at the beginning of the project, which meant that some parts of the material from the last weeks almost had to be implemented or fitted retroactively to fit with the problem.
In general we tried to distribute out the work roughly equally, with different people having the main responsibility for the different parts of the projects, in order to best play to our different strengths. But we also made sure we sat together when working so we always had the opportunity to discuss the different problems with eachother. From a programming aspect, we used a combination of Git for version control and the extension "Code with me" which allowed us to work together live and in that way eaiser follow the development. Overall it was a challenging project, due to size and complexity of the project, but also a very rewarding, as you really learn the course material by actually implementing it all into a real project.

6 Appendix

6.1 Git repository

The URL to our GitRepository: `https://github.com/WPeytz/SoftwareEngineeringProjekt`