David Hu - 260986407
James Willems- 260921223
…

1. Summary of Deliverables

The deliverable for part B of this project consists of feature files containing the user stories, step definitions to match each step in said user stories, a cucumber tests runner and helper methods to keep the tests short.

Each user story is written with the purpose of what a student using the todo manager application would use, meaning each story is related to a potential scenario an everyday student might have, such as categorizing projects or keeping track of remaining tasks.

That being said, the tests are heavily related to what the API provides in the application, and each user story explores a functionality provided in the API itself by using the given endpoints as end goals. For example, a provided endpoint could be to list all the remaining todos in a project, then a user story based on it would be a student who inquires about all the tasks remaining in a specific project.

All feature files are structured as follow: First the

2. Description of structure of story test suite

Each test suite is designed differently according to what the user story is. The user stories written by our team consists of a few distinct types of suite. One of them would be adding a class of one type to the subclasses of another. As an example, adding categories to a project is implemented by creating a project, then adding the category to said project and verifying that the project was added.

The suite would therefore have a structure which is very in line with the description above. Given a created project, when I add a category to said project, then the project shall have the new category in its category list. However, there are other possible scenarios for each user story. As an example, if the user tries to add a project to a category which does not exist, then the test verifies that the program indeed returns an error code. Our team has chosen to do cucumber tests, which allows us to easily create multiple examples of each scenario.

The test suite above can be implemented for every cross class POST operation, as the logic is the exact same, but there are other possible user stories which are related to different operations, such as getting all the referenced classes of a class.

Another example of a story suite would be a student who needs to see all the categories of the project they are working on. This suite will be different from the one already discussed as it requires the Given step to set up a project with multiple existing categories.

Then, when the list of projects is fetched, the final Then step is required to verify both categories are indeed in the project instead of simply checking if a category was added. Some alternate scenarios include fetching the list of categories from a non-existent project or the categories list having some empty titles.

Furthermore, POST and GET are very different from DELETE. An user might want to delete a project from a category once the project is finished. It is similar to the GET request since it requires the Given step to set up, but unlike the GET, the When step deletes the created association, and the final Then step verifies it is removed. It does so by iterating over all existing associations and verifying the deleted one is not present.

Table showing all our User Stories:

Abe

User Story	Given	And	When	Then
As a student, I add a task to a class to-do list, so I don't forget it	POST project GET projects	POST todo GET todos	POST todo/id/taskof	GET todo/id
As a student, I categorize my tasks as Urgent when they are urgent, so I can prioritize them	POST todo POST category GET todos GET categories	GET categories/id	POST todo/id/category	GET todo/id
As a student, I mark a todo as complete when it is done, so I can track my progress	POST todo GET todos		POST todo/id	GET todo/id
As a student, I delete a category that I no longer use, so I can forget about it	POST category GET categories		DELETE category/id	GET categories
As a student, I delete a todo if it was cancelled, so I don't get confused	POST todo GET todos		DELETE todo/id	GET todos

James

As a student, I want to modify tasks description so they can better illustrate what I want to do	POST todo GET todo		POST todo/id	GET todo/id
As a student, I want to delete projects I no longer do, so I can focus on another	POST project POST todo POST projects/id/tasks		DELETE project/id	GET project
As a student, I want to get all todos in a project that are incomplete, so I can work on them	POST todo POST project POST todo	POST todo/id/tasksof POST todo/id/tasksof		GET project/id/tasks
As a student, I want to delete todos in a project that are complete, so that I can focus on incomplete ones	POST todo POST project POST todo	POST todo/id/tasksof POST todo/id/tasksof	DELETE project/id/tasks/id	GET project/id/tasks
I want to create tasks to help figure out what to do			POST todo	GET todo/id

David

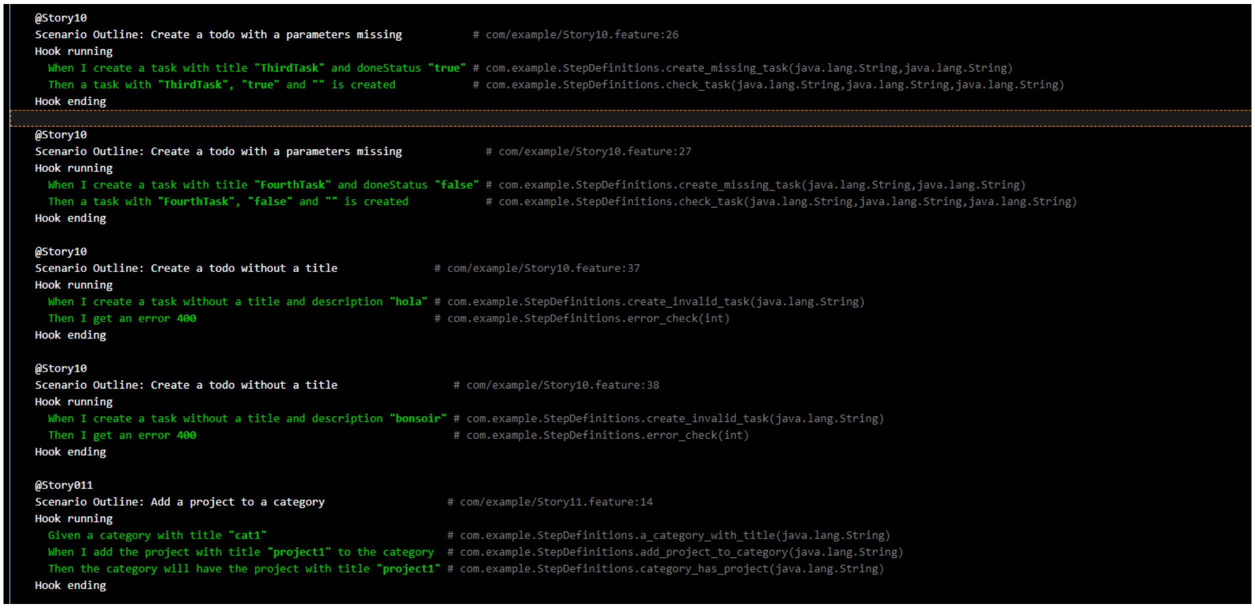
User Stories	Given	And	When	Then
As a student, I want to add projects to a category to classify each projects	POST category POST project GET categories GET projects	GET category/id	POST category/id/projects	GET project/id
As a student, I want to delete projects in categories once they are completed	POST project POST category POST project	POST category/id/projects POST category/id/projects	DELETE category/id/projects/id	GET category/id/projects
As a student, I want to get all the projects in a category to work on projects of the same category	POST project POST category POST project	POST categories/id/projects POST categories/id/projects		GET categories/id/projects
As a student, I want to delete todos in a category once the task is completed	POST todo POST category POST todo	POST category/id/todos POST category/id/todos	DELETE category/id/todos/id	GET category/id/todos
As a student, I want to get all the categories associated to a project to see what the project is classified as	POST category POST project POST category	POST projects/id/categories POST projects/id/categories		GET projects/id/categories

Nikhil

User Story	Given	And	When	Then
	POST project GET projects	POST todo GET todos	POST todo/id/taskof	GET todo/id
	POST todo POST category GET todos GET categories	GET categories/id	POST todo/id/category	GET todo/id
	POST todo GET todos		POST todo/id	GET todo/id
	POST category GET categories		DELETE category/id	GET categories
	POST todo GET todos		DELETE todo/id	GET todos

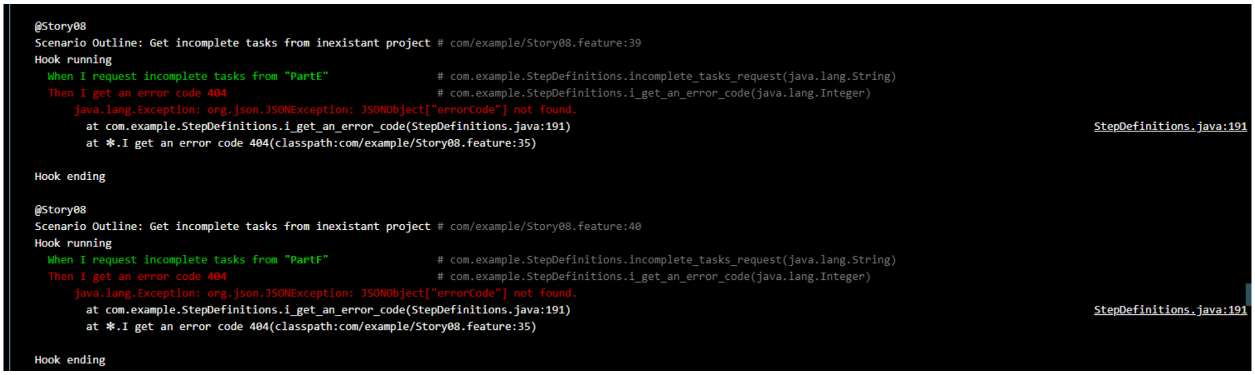
3. Description source code repository

   Our Features and tests are located in the ecse429-partb folder under src/test. Sadly we weren't able to make them run with Maven despite our attempts, so to run them go to RunCucumberTest and run the program. In VSCode, you can see our tests passing in the Debug Console, where everything is green except a few tests designed to fail due to a bug we discovered and that will be explained later. Before each Scenario and after them, we run hooks to initialise the API and then shutting it down, ensuring no data from previous tests persists. This however does mean you cannot run the API already while running the tests, as this will cause the data to stay and make our tests fail. As for the random run, we couldn't figure out a way of doing it, our only claim that the various scenarios don't rely on each others is the fact we wipe clean the database after every instance, ensuring independence between cases.



```
@Story10
Scenario Outline: Create a todo with a parameters missing          # com/example/Story10.feature:26
Hook running
  When I create a task with title "ThirdTask" and doneStatus "true" # com.example.StepDefinitions.create_missing_task(java.lang.String,java.lang.String)
  Then a task with "ThirdTask", "true" and "" is created            # com.example.StepDefinitions.check_task(java.lang.String,java.lang.String,java.lang.String)
Hook ending

@Story10
Scenario Outline: Create a todo with a parameters missing            # com/example/Story10.feature:27
Hook running
  When I create a task with title "FourthTask" and doneStatus "false" # com.example.StepDefinitions.create_missing_task(java.lang.String,java.lang.String)
  Then a task with "FourthTask", "false" and "" is created            # com.example.StepDefinitions.check_task(java.lang.String,java.lang.String,java.lang.String)
Hook ending

@Story10
Scenario Outline: Create a todo without a title               # com/example/Story10.feature:37
Hook running
  When I create a task without a title and description "hola" # com.example.StepDefinitions.create_invalid_task(java.lang.String)
  Then I get an error 400                                     # com.example.StepDefinitions.error_check(int)
Hook ending

@Story10
Scenario Outline: Create a todo without a title                  # com/example/Story10.feature:38
Hook running
  When I create a task without a title and description "bonsoir" # com.example.StepDefinitions.create_invalid_task(java.lang.String)
  Then I get an error 400                                        # com.example.StepDefinitions.error_check(int)
Hook ending

@Story011
Scenario Outline: Add a project to a category                  # com/example/Story11.feature:14
Hook running
  Given a category with title "cat1"                           # com.example.StepDefinitions.a_category_with_title(java.lang.String)
  When I add the project with title "project1" to the category  # com.example.StepDefinitions.add_project_to_category(java.lang.String)
  Then the category will have the project with title "project1" # com.example.StepDefinitions.category_has_project(java.lang.String)
Hook ending
```

Example output of our tests



```
@Story08
Scenario Outline: Get incomplete tasks from inexistant project # com/example/Story08.feature:39
Hook running
  When I request incomplete tasks from "PartE"                 # com.example.StepDefinitions.incomplete_tasks_request(java.lang.String)
  Then I get an error code 404                                 # com.example.StepDefinitions.i_get_an_error_code(java.lang.Integer)
    java.lang.Exception: org.json.JSONException: JSONObject["errorCode"] not found.
      at com.example.StepDefinitions.i_get_an_error_code(StepDefinitions.java:191)                                  StepDefinitions.java:191
      at ✱.I get an error code 404(classpath:com/example/Story08.feature:35)

Hook ending

@Story08
Scenario Outline: Get incomplete tasks from inexistant project # com/example/Story08.feature:40
Hook running
  When I request incomplete tasks from "PartF"                 # com.example.StepDefinitions.incomplete_tasks_request(java.lang.String)
  Then I get an error code 404                                 # com.example.StepDefinitions.i_get_an_error_code(java.lang.Integer)
    java.lang.Exception: org.json.JSONException: JSONObject["errorCode"] not found.
      at com.example.StepDefinitions.i_get_an_error_code(StepDefinitions.java:191)                                  StepDefinitions.java:191
      at ✱.I get an error code 404(classpath:com/example/Story08.feature:35)

Hook ending
```

Tests failing due to bug

4. Describe findings of story test suite execution

While working on User Story #8, we discovered that even if a project didn't exist, the API would still send us back a collection of todos regardless of the filters we added to the query, this is why the two error paths tests fail when they should have an error 404: Not Found.