Developer guide Digi-Sm

Benjamin Richter

April 2023

Contents

1 Introduction

This is the developer guide for the Digi-Sm web app. The guide is meant to support any developer wanting to understand the code in order to be able to expand upon it or adapt it to their needs. If you're looking to understand how to use the existing features you'll be better off checking the user guide. This guide will go over the structure of the code, the steps to installing and running the code on your machine, general information on the working of the website and finally the packages used to run the website.
The main idea behind this guide is to allow programmers to get a grip on the codebase quickly and to be able to adapt it to their needs.

2 What is Digi-Sm

Digi-SM, short for Digital Scrum Master, is a powerful and intuitive website designed to help you and your team develop projects and implement the Agile methodology with ease. With Digi-SM, you can effectively manage all aspects of your project development process, from creating user stories to organizing sprints, tracking progress and visualising this data through graphs and diagrams.

One of the main benefits of Digi-Sm is its simple and user friendly interface. You can easily create an account and get straight to work from any device connected to the internet. This allows your team to spend less time on planning and thus more time on getting your project on track!

Digi-SM's project management features are designed to help you and your team stay on track and organized throughout the development process. You can create and input all of your user stories at once, making it easy to see at a glance what needs to be done where and when.

Whether you're just starting out or looking to streamline your existing processes, Digi-SM can help you and your team achieve your project development goals.

3 Technology used

3.1 Back end

The backend of the website is mainly built on the flask web development framework. This allows for the code base of the website to be written in python and facilitates the creation and management of different routes.
Further flask integration packages are used in order to simplify the management of forms, login, database management and session cookies. These are described more thoroughly in the packages used section of the guide.

3.1.1 Data Storage

The website requires the storage of a lot of data used to create, manage and work on projects. Furthermore user data needs to be backed up and linked to a persistent table containing all the projects the user and his team are working on. All of this runs on an SQLAlchemy database, an object relational mapper, allowing to facilitate the manipulation of the database within the flask app. This database can be accessed through the following file:
`/code/Digi-sm/my_app/models.py`
The model was simplified in order to be easily manipulated and allow access to the tables with simple object oriented like calls such as:
`ProjectRequest.query.filter_by(user_id=current_user.id).all()` And with the most complicated structures being association tables accessible with simple filters such as:
`db.session.query(asso_table.c.project_id).filter(asso_table.c.user_id == current_user.id).all()` which allows to filter the association between the users and the project they're associated to. A real use example of this can be found at the call of the `amountOfProjects` variable within the main route. Click here to find the graph of the database 15.

3.2 Front end

The front was made thanks to a mixture of html and css. The logic behind it is assisted by Jinja and driven thanks to javascript which powers features like the pi-chart or the burndown chart thanks to ChartJs. All the imports necessary for the website can be found in the root template which is called home.html and can be found at: `/code/Digi-sm/my_app/templates/home.html`
This includes packages such as bootstrap, awesome font and most importantly jquery which is used within the frontend with ajax in order to communicate with the backend.

4 Structure of the Program

```
|
|
|__ myapp
|   |__ static
|   |   |__ Main.js
|   |   |__ css
|   |__ templates
|   |   |__ charts.html
|   |   |__ etc
|   |   |__ userOneProject.html
|   |__ __init__
|   |__ forms
|   |__ login_manager
|   |__ models
|   |__ nonFlaskFunctions
|   |__ routes
|__ tests
|   |__ unit
|       |__ __init__
|       |__ test_model
|       |__ test_model
|__ config
|__ data.db
|__ README
|__ requirements
```

5 Packages used

The following is the list of packages and their uses in the flask application

- Flask - used to create the routes thus being the backbone
- Flask-WTF - used to create forms and communicate data back to the server
- Flask-Login - to ensure the password is safe and hashed
- Flask-SQLAlchemy - to run the database and manipulate tables as objects
- Flask_session - to use cookies to save data such as information related to project the user is working on and keeping it while he's on the website

6 Installation and Setup

A basic graphic interface was created in order to facilitate the launching of the website. Thus in order to launch the website follow these steps:

1. Once in the folder of the app navigate to the `/bin` folder
2. Make sure python is installed and added to your environment variables (if not you can do so here)
3. Click on the launchServer.bat file. This will launch a basic GUI allowing you to create a virgin database, load the existing database or load a dummy database to test the website. Keep in mind that creating or loading the dummy database will overwrite any existing database.

If this works successfully the program will automatically download all the requirements necessary to run the server. If that isn't the case and you're struggling to launch the server through the GUI then you can follow these steps too:

1. Once in the folder of the app navigate to the `/code/Digi-sm/` folder
2. When you're there you'll need to start by making sure you have python correctly installed and in your path, if not you can download it here).
3. Then you can proceed by installing all the modules required for the program by executing the `pip install -r requirements.txt` command.
4. Once the requirements are downloaded you can launch the website with these commands:
   (a) `python run.py` to launch the website normally and load the existing database in the folder (if non-existent it'll create one by default).
   (b) `python run.py -create` to launch the website and create a new database, careful this will overwrite any existing database.
   (c) `python run.py -dummy` to launch the website and load some dummy data within the database to test the website.

This should get you on your way to installing and setting up your website! Note that the code in its current state still runs with the flask debugger activated to facilitate an easier debugging. If you wish to disable debugging simply head into the run.py folder and set the debug variable at the end of the file to False.

7 Routes

This section provides a short overview of the existing routes types from a top down perspective. Meaning starting from the highest object a Project all the way down to individual user stories. For a more detailed description of each routes individually please refer to the specifications within the code.

7.1 Main

This is the main route which loads all the required data for a project and a sprint you're looking at, this is the heart of the website and most other routes redirect towards this route once they've finished their tasks.

```
@app.route('/' , methods=['GET', 'POST'])
```

7.2 Projects

These routes all serve to create and manage the projects a user wishes to create. These projects will in turn include a series of sprints and users in order to build the project.

```
@app.route("/createProject", methods = ['GET', 'POST'])
@app.route("/finishProject/<int:project_id>")
@app.route("/choose_project/<int:project_id>")
@app.route("/redirectProjects")
```

7.3 Sprints

These routes all serve to manage the sprints and their data. This includes simply loading the sprint data within the session cookie when interacting with the timeline all the way down to creating entire new sprints.

```
@app.route("/load_sprint/<int:sprint_id>", methods = ['GET', 'POST'])
@app.route("/shift_sprint/", methods = ['GET', 'POST'])
@app.route("/create_sprint/", methods = ['GET', 'POST'])
@app.route("/update_order", methods = ['GET', 'POST'])
```

7.4 User stories

These routes are made for the management of the user stories. They allow for the creation of new user stories but also for the editing of these user stories once they've been attributed to a sprint if, for example, the poker score was deemed to be misjudged.

```
@app.route("/newTask", methods=['GET', 'POST'])
@app.route("/delete/<int:task_id>")
@app.route("/check_uncheck/<int:task_id>")
@app.route("/modify/<int:task_id>", methods=['GET', 'POST'])
```

7.5 User data

These routes allow for the creation and editing of everything related to the user. Furthermore these are the routes used to use admin rights or to add users to projects in different ways.

```
@app.route("/profile", methods = ['GET', 'POST'])
@app.route('/userOnProject', methods=['GET', 'POST'])
@app.route('/addUserProjectByUsername', methods=['GET', 'POST'])
@app.route('/addUserProjectButton', methods=['GET', 'POST'])
@app.route("/friends", methods = ['GET', 'POST'])
@app.route('/team', methods=['GET', 'POST'])
@app.route("/bloc_user/<int:user_id1>")
@app.route("/change_group/<int:user_id1>")
@app.route('/admin', methods=['GET', 'POST'])
```

7.6 Registration and login

These routes are used in order to register, login and logout users in a safe manner.

```
@app.route('/register', methods=['GET', 'POST'])
@app.route('/login', methods=['GET', 'POST'])
@app.route('/logout')
```

7.7 Notifications

These routes are used in order to register, login and logout users in a safe manner.

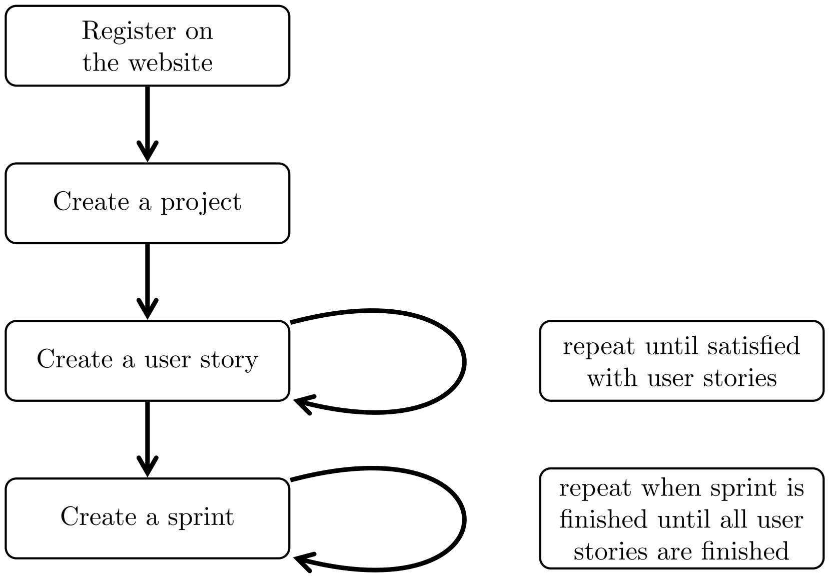
```
@app.route('/notification', methods=['GET', 'POST'])
```

7.8 Help messages

These are the routes used to load in the help messages spread around the website. The icons can be found within the html code and redirect towards this route giving it an id. This id is then stored in the session cookies and loaded through the redirect. The use of this id can be seen in the home.html if else statement.

```
@app.route('/helperMessage/<int:helpMessageId>')
```

8 Workflow


Register on the website
Create a project
Create a user story
repeat until satisfied with user stories
Create a sprint
repeat when sprint is finished until all user stories are finished

9 Sprints used to develop this project

If you wish to observe the evolution of the project you can go in the legacy folder of the project to see the different sprints that made up this project. Furthermore within the planning folder you'll be able to find a markdown file used to plan and organise the different sprints and the user stories contained within these sprints. Furthermore you can also refer to the installation and setup guide in order to launch the website with the pre-prepared dummy data containing all the user stories and sprints that were used to make this website into reality.

10 How to expand on the code

The project is setup in a way so that future developers wishing to expand on the code can proceed in a number of different ways.

10.1 Create New routes

In order to create new routes you can head into the routes folder containing the flask code which can be found at:
`/code/Digi-sm/my_app/routes.py`
Routes can be used either to prepare data from the backend and pass it on to a new html template or to handle backend work.

10.1.1 Routes to handle backend work

Routes to handle things purely in the backend (such as update_order()) are called by the frontend and allows for example to update information the backend. These routes can be recognised by the fact that they do not redirect towards another template but simply return a http code to indicate the success or failure of the request. Within these routes you can do anything you wish on the backend, such as changing data within the session cookies or updating the database.

10.1.2 Routes to load a Html page

Routes to handle the loading of html pages are similar to the last point, with the exception that their main use is to load and format data which the page will use. Furthermore the return of the function is a render template call with the data passed as arguments. This data can then be accessed by the front end thanks to the inbuilt Jinja calls which allow to access data such as elements of a list called elem in this way: `{{elem[0]}}`

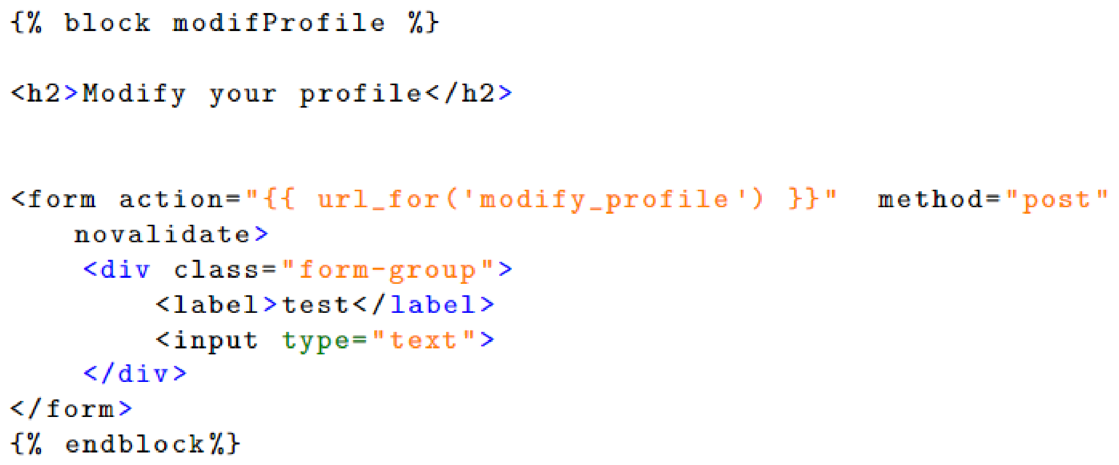
Both of these kinds of routes are called by the frontend in two different ways, the first one being a simple `href="/routeToCall"` on buttons and the second one being an ajax call to the backend such as:

```
jQuery.ajax({
    type : 'POST',
    data : {'data':JSON.stringify(userS)},
    url : "/create_sprint",
    success: function () {
        sessionStorage.clear();
        location.href = '/';
    }
  });

}
```

10.2 Create New Html Pages

If you wish to create a new page which contains the same navigation bar as the rest of the application, you can simply create your new html file and encase it within a block like this.

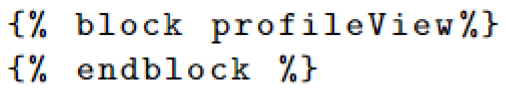


```
{% block modifProfile %}

<h2>Modify your profile</h2>


<form action="{{ url_for('modify_profile') }}"  method="post"
    novalidate>
     <div class="form-group">
         <label>test</label>
         <input type="text">
     </div>
</form>
{% endblock%}
```

Once this is done you can simply call add it to the home.html file like this:



```
{% block profileView%}
{% endblock %}
```

And if within the flask code you render the new html template you've created it'll be loaded in with the navigation bar at the top.

Alternatively if you don't wish to have the navigation bar available you can simply just load the name of your template within the return render template of your python route.

10.3 Expand purely on backend functionalities

In order to expand on backend functionalities you can simply use the:
`/code/Digi-sm/my_app/nonFlaskFunctions.py`
file in order to implement whatever new function you would like to use within the appropriate route. To use them simply add them to the import at the top of the routes.py file.

11 Strengths and limitations

This section will briefly discuss the perceived strengths and weaknesses of the website and what can be done to potentially address these issues.

11.1 Strengths

The advantage of the website being built on flask is that python is an easy to grasp language. The code and names of variables are very expressive and thus allow you to grasps the logic behind the routes and websites quickly. Furthermore plugins such as `SQLite` on visual studio code enables for a quick and easy visualisation of the database in order to observe it as it evolves or to debug issues by looking directly at the effect that the code has on it.

11.2 Limitations

One of the main limitations of the website is the way the database is setup. Whilst its fully functional and works as intended, the fact of the matter is that it's built directly into python and could be found to be less than optimal for larger scale use.
This being said SQLAlchemy is made to be used with lighter framework database language tools such as SQLite as well as more established languages such as MySQL or PostgresSQL. Meaning that if your intention is to release this website for a wider audience, all the functions and calls made within the routes should still be fully functional and you would only need to create your database and connect to the website.

12 Testing

The test file is already fully setup and runs thanks to the PyTest module.

12.1 Launching the test suite

In order to launch the test simply navigate to the Digi-sm folder: `/code/Digi-sm` and if you have PyTest installed simply launch the following command: `pytest` If you do not have PyTest installed you can do so with the following command: `pip install pytest`

12.2 Adding New Tests

If you wish to add new tests to the program simply go into the test folder found here: `/code/Digi-sm/tests/unit` Here you can add some tests within the existing file or simply create a new file to test further functionalities.

13 Troubleshooting

If you haven't fully deployed the website then the debugging feature within flask should still be active and facilitate debugging by indicating wherever the error has occured within the code. This will allow you to spot where within the routes of more genreally in the python code the error has occured. If an error has occured in the front end, simply identify the source problematic javascript function thanks to the dubugger and don't be scared to get to grip with the code by including console.logs to observe the behaviour of the code.

14 FAQ

14.1 Features of the website

14.1.1 Can I export the analytics generated by the website?

Not yet- Digi-SM offer a variety of metrics to track the advancement of your project. These have been simplified thanks to the two main visuals available from the front page and are easily accessible once you've loaded your project, but cannot be exported yet. This can easily be achieved though as the functions which calculate and prepare all the metrics can be found within the non-flask python functions.

14.1.2 Can I mark a sprint as finished?

Yes and no- your sprints are done if all the user stories within them are marked as Done. You can see this by loading the pie chart which allows you to quickly visualise what user stories are left to accomplish, are still being worked on or are finished. So there is a way to visualise it yes but no there is no directly accessible metric. If you wish to do so you could simply add a button to the html and either a new attribute to the sprint table in the database or simply create a route which will mark all user stories as done and a filter checking if that's the case in the main route.

14.1.3 Can I change my password?

Yes- Once on the homepage press the "Profile" button in the top right corner of your screen. Once on the profile page you'll see your information is loaded. The two fields at the bottom is where you can enter your new password and its confirmation in order to change it.

14.2 Learning to be an agile developer

14.2.1 Where can I get more information about the agile methodology?

The agile methodology is very flexible and usually everyone has their own variant of the technique. This being said you can learn the template of the agile methodology through the tutorial spread throughout the website. Just lookout for the little blue question marks. And if you need any more information you can head to https://www.atlassian.com/agile which is a wonderful source of information.

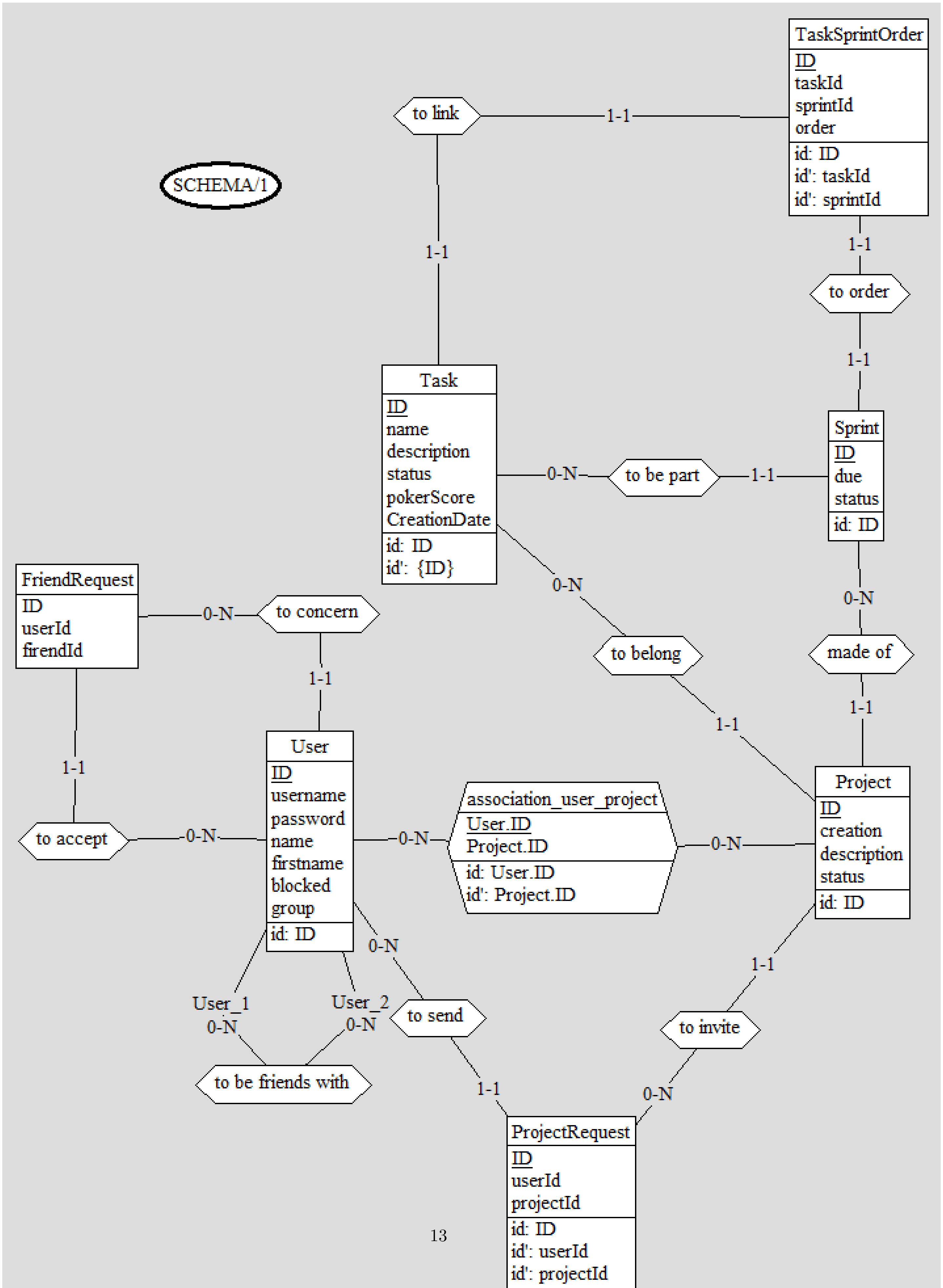
14.3 Contact information

14.3.1 How can I contact support if I have any questions or issues with Digi-SM?

This is an open-source project realised for a bachelor project. This means that there is no team which will be able to answer quickly but you can reach me at benjamin.richter@student.unamur.be and I'll come back to you as quickly as I can.

15 Glossary of terms

1. Agile methodology: An iterative approach to project management that emphasizes collaboration, flexibility, and customer satisfaction.
2. Backlog: A list of prioritized tasks or user stories that need to be completed in a project.
3. Burndown chart: A visual representation of the amount of work remaining in a sprint or project, typically plotted against a timeline.
4. Scrum: A framework for Agile project management that emphasizes teamwork, accountability, and regular progress reviews.
5. Sprint: A time-boxed period (usually 1-4 weeks) during which a team works to complete a set of tasks or user stories.
6. User story: A short, simple description of a feature or requirement from the perspective of an end user.


SCHEMA/1
TaskSprintOrder
ID
taskId
sprintId
order
id: ID
id': taskId
id': sprintId
to link
1-1
1-1
1-1
to order
1-1
Task
ID
name
description
status
pokerScore
CreationDate
id: ID
id': {ID}
0-N
to be part
1-1
Sprint
ID
due
status
id: ID
0-N
to belong
0-N
made of
1-1
1-1
FriendRequest
ID
userId
firendId
0-N
to concern
1-1
1-1
to accept
0-N
User
ID
username
password
name
firstname
blocked
group
id: ID
0-N
association_user_project
User.ID
Project.ID
id: User.ID
id': Project.ID
0-N
Project
ID
creation
description
status
id: ID
0-N
User_1
0-N
User_2
0-N
to be friends with
to send
1-1
1-1
to invite
0-N
ProjectRequest
ID
userId
projectId
id: ID
id': userId
id': projectId