Official documentation

Idea: Media player
Leader : Gerald Prendi
Members:
- Ramazan Tafa
- Lidia Morariu
- Paola Guma

Road map:	**Dates:**
1. Project kick-off :	05.11.2020 – 18.11.2020
✔ Proposal	12.11.2020
✔ Start-up	18.11.2020
✔ Versioning system	15.11.2020
✔ Communication tools	08.11.2020
✔ Project Homepage	13.11.2020
2. Design:	18.11.2020 – 25.11.2020
✗ GUI	25.11.2020
✗ Back-end	25.11.2020
3. Implementation :	26.11.2020 – 05.12.2020
4. Debugging & Testing	06.12.2020 -- 08.12.2020
5. Final :	09.12.2020 – 11.12.2020
✗ Final Release	09.12.2020
✗ Distribution	10.12.2020
6. Presentation :	10.12.2020 – 18.12.2020
✗ Presentation ready	14.11.2020

Work Groups:
- Developers :
  In charge of writing code, creating test cases and writing proper documentation. All the group members are included.
- Graphic Designers :
  In charge of the design, the GUI and the look & feel of the program. Able to use graphics' tools to generate a complete UI.
  Members: Paola & Lidia
- Back-end Dev :
  In charge of the under the hook technologies and code. Responsible of processing data, interaction with the file system and maintaining the structure of the "databases ".
  Members: Gerald & Ramazan

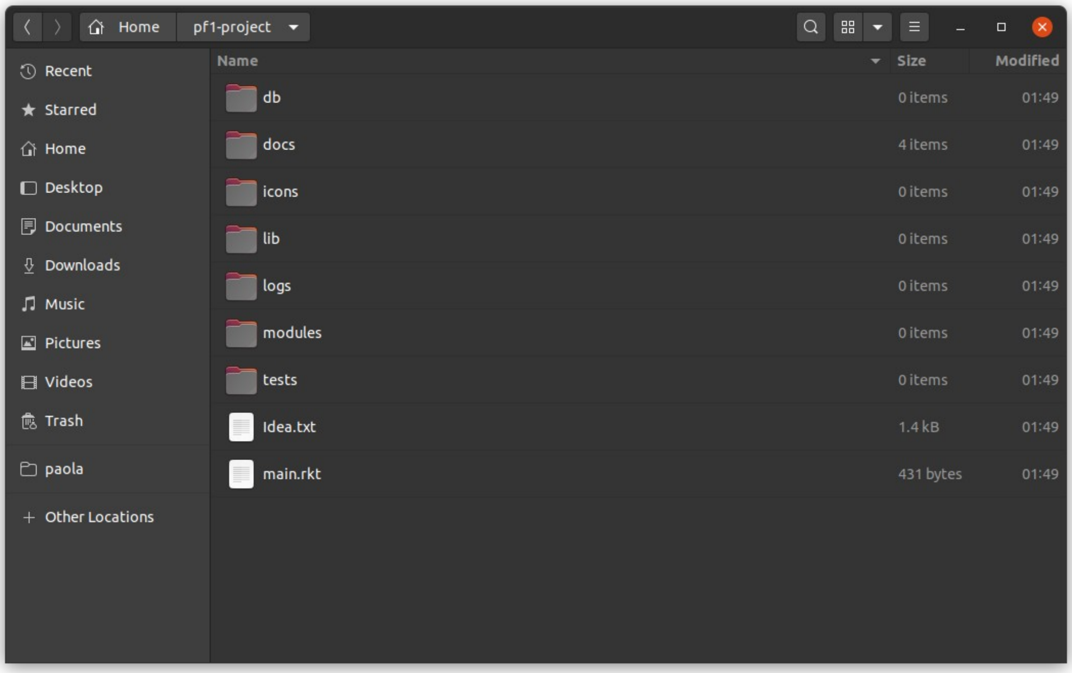
- DevOps :
  In charge of maintaining the infrastructure, debugging, packing and distributing the product. Takes care of the versioning system, communication tools and the project homepage.
  Members: Gerald

Roles :
Gerald :
    Developer, Back-end Dev, DevOps
    Lead and check the progress of the group members.
    Sets up the infrastructure (versioning system, communication tools)
    Helps in the coding and design of the back-end.
    Sets up deadlines and milestones.
    Polishes and cleans the repository.
Ramazan:
    Developer, Back-end Dev
    Designs and codes the back-end, writes tests too.
    Helps in the coding of the GUI, some parts of it.
    Debugs and tests the program in the last phase.
Lidia :
    Developer, Graphics Designer
    Delivers complete UI with full documentation.
    In charge of the progress of designing the UI.
    Codes and writes test cases.
    Helps with the presentation in the final phase.
Paola:
    Developer, Graphics Designer
    Idealizes and designs the UI by documenting every step.
    In charge of the progress of coding the GUI.
    Debugs the program by writing test cases and applying fixes.
    Designs the presentation in the final phase.

Versioning System:
We are using SubVersioN (SVN) because of the familiarity with the tool. It is hosted on the atelier's web server.
To set up the repository:
    You should have SVN installed and proper permissions on the project directory.
To checkout the repository use the following command:

```
svn checkout
ssh+svn://your_username@atelier.inf.usi.ch/home/prendg/svn/pf1-
project
```

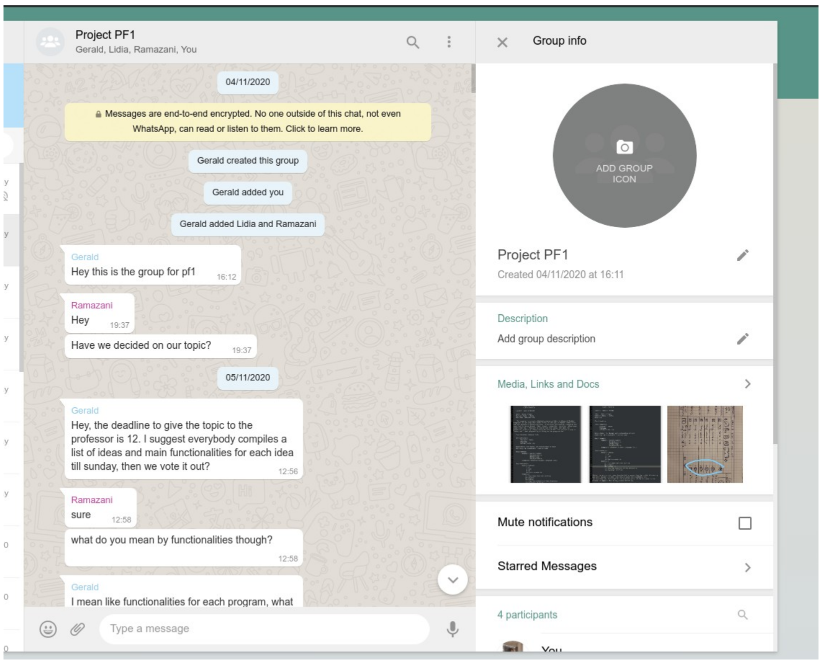
Repository:



Name	Size	Modified
db	0 items	01:49
docs	4 items	01:49
icons	0 items	01:49
lib	0 items	01:49
logs	0 items	01:49
modules	0 items	01:49
tests	0 items	01:49
Idea.txt	1.4 kB	01:49
main.rkt	431 bytes	01:49

- We thought it was better, that the work Repository to be a multi file/folder project, in order for everybody to work in parallel.
- Explanation: ‘main.rkt’ is the main executable file, everything will come down to main(It will one have the implementation of big-bang)
- ’lib’ folder is the folder where libraries will be installed, we intend to download and put libraries we usually require or install using ’raco’ in there.
- ’modules’ folder will have files: ’modules’ we created ourselves and will link them to main
- ’docs’ will have the documentation
- ’db’ will host the info files, in form of database(we will not implement a database, just plain text files)
- ’icons’ is a folder for icons needed
- ’logs’ will have error messages or known bugs, that will be fixed
- ’test’ folder will have test files, for each main modules

Communication tools:
We use several communication platforms such as whatsapp, Microsoft Teams and Notion.
Whatsapp:


Project PF1
Gerald, Lidia, Ramazani, You

04/11/2020

🔒 Messages are end-to-end encrypted. No one outside of this chat, not even WhatsApp, can read or listen to them. Click to learn more.

Gerald created this group

Gerald added you

Gerald added Lidia and Ramazani

Gerald
Hey this is the group for pf1 16:12

Ramazani
Hey 19:37
Have we decided on our topic? 19:37

05/11/2020

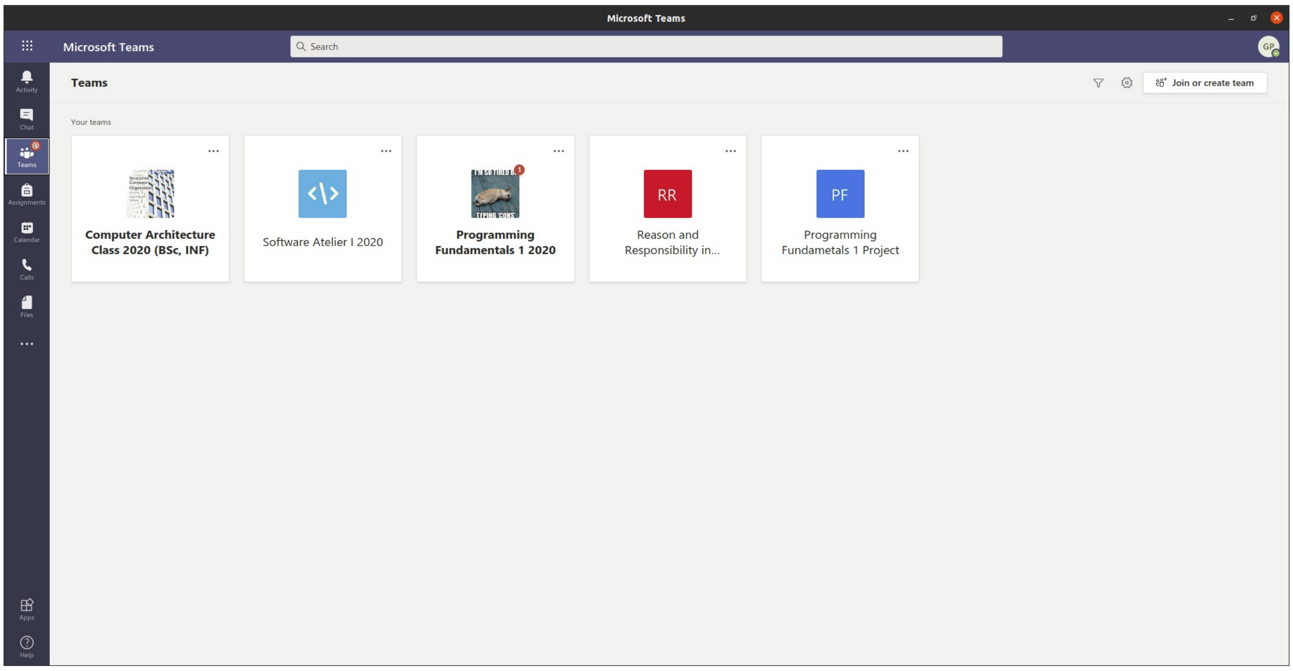
Gerald
Hey, the deadline to give the topic to the professor is 12. I suggest everybody compiles a list of ideas and main functionalities for each idea till sunday, then we vote it out? 12:56

Ramazani
sure 12:58
what do you mean by functionalities though? 12:58

Gerald
I mean like functionalities for each program, what

Group info
Project PF1
Created 04/11/2020 at 16:11
Description
Add group description
Media, Links and Docs
Mute notifications
Starred Messages
4 participants

Microsoft teams:



Teams

Your teams

Computer Architecture Class 2020 (BSc, INF)

Software Atelier I 2020

Programming Fundamentals 1 2020

Reason and Responsibility in...

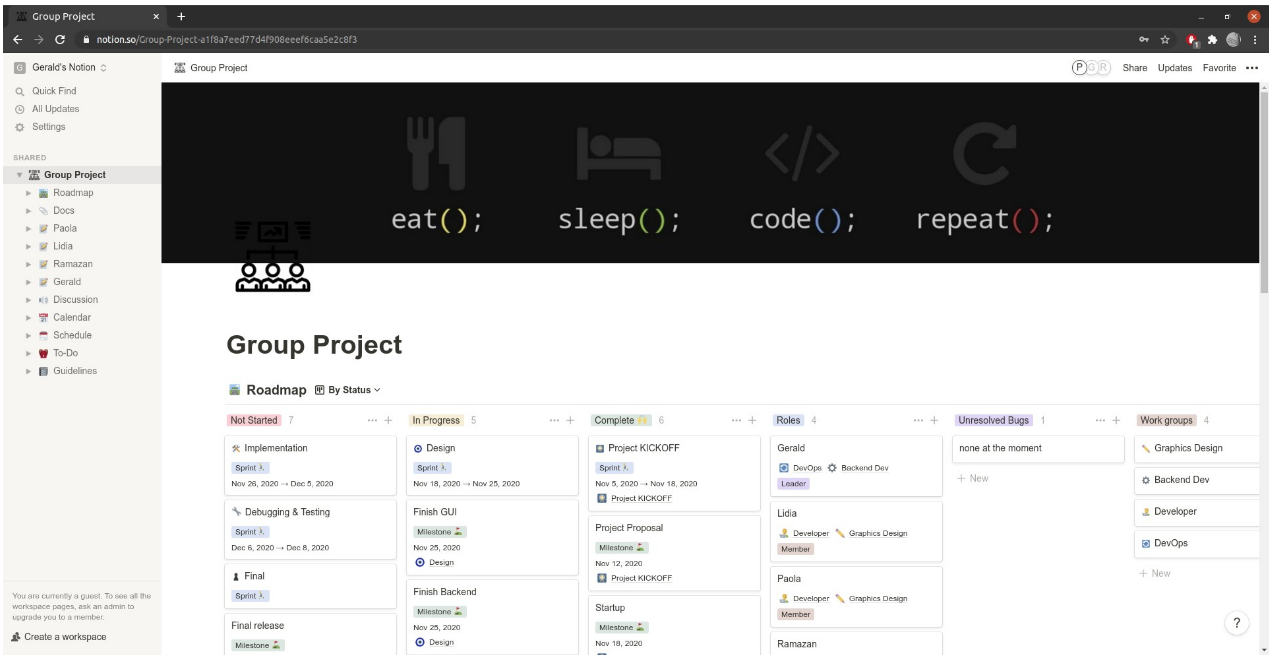
Programming Fundametals 1 Project

Notion:
(An image of notion will not be displayed because it will be part of the homepage.)

Project Homepage

Can be accessed by the link :
http://player.csproject.org

It points to the notion page where the project is hosted.



You are currently a guest. To see all the workspace pages, ask an admin to upgrade you to a member.

Group Project

Roadmap

Not Started 7	In Progress 5	Complete 6	Roles 4	Unresolved Bugs 1	Work groups 4
Implementation Sprint Nov 26, 2020 → Dec 5, 2020	Design Sprint Nov 18, 2020 → Nov 25, 2020	Project KICKOFF Sprint Nov 5, 2020 → Nov 18, 2020 Project KICKOFF	Gerald DevOps Backend Dev Leader	none at the moment	Graphics Design
Debugging & Testing Sprint Dec 6, 2020 → Dec 8, 2020	Finish GUI Milestone Nov 25, 2020 Design	Project Proposal Milestone Nov 12, 2020 Project KICKOFF	Lidia Developer Graphics Design Member		Backend Dev
Final Sprint	Finish Backend Milestone Nov 25, 2020 Design	Startup Milestone Nov 18, 2020	Paola Developer Graphics Design Member		Developer
Final release Milestone			Ramazan		DevOps

- ◆ Roadmap is the main part where the progress made is visible .
- ◆ Docs is where you can find all the documentation related to our project.
- ◆ There is a file for each group member where it is necessary to write and upload ideas and progress constantly.
- ◆ Discussion is where the new ideas and topics are hosted.
- ◆ We have created a calendar with all the important dates and milestones of the project advancement.
- ◆ The section To-Do refers to the work needed to be completed immediately.
- ◆ And lastly Guidelines which is used for all the rules that **must** be followed during the execution of this project.

Coding Guidelines:
The standard language is Advanced Student Language(ASL)
You are allowed to require these modules:

- 2htdp/image
- 2htdp/universe
- 2htdp/batch-io
- racket/strings
- racket/system
- racket/base

If you want to use a different module, ask the group leader.
Define VARIABLES always in upper-case.

Define Variables and Functions in a meaningful way, not just VARIABLE.

Define the DataStructure that you used in the implementation

For each problem, formulate the solution in a piece of paper first, then implement it following the steps you thought of.

Make two copies of the files that you are working on, one for testing and one where you deliver the working copy.

Make sure to commit. Follow two simple rules:
-Always commit a working copy, which is tested and ready.
- Write meaningful commit messages.

Write comments on every step of the way,make it understandable (you don't have to write header and template).

Testing will be done on separate files, and will be saved on a separate folder.
All modules, that you used it's better to get them from the folder "~/usr/racket/collects" and put them in the lib folder. *(ask Gerald for help, if you have any problem).*

When writing your own modules:
-For every function that you want to be accessed globally (by other functions):
-Use "(provide name-of-function)" and the function can be called.
-Use "(require path/to/file)" to be able to use the function you created.
-There will be a global.rkt file in the modules folder, where all the global functions will be documented, and called, and anybody who provides must also document it there.
TIP: always update the repository, when you start working and always check the global.rkt file.

How to test

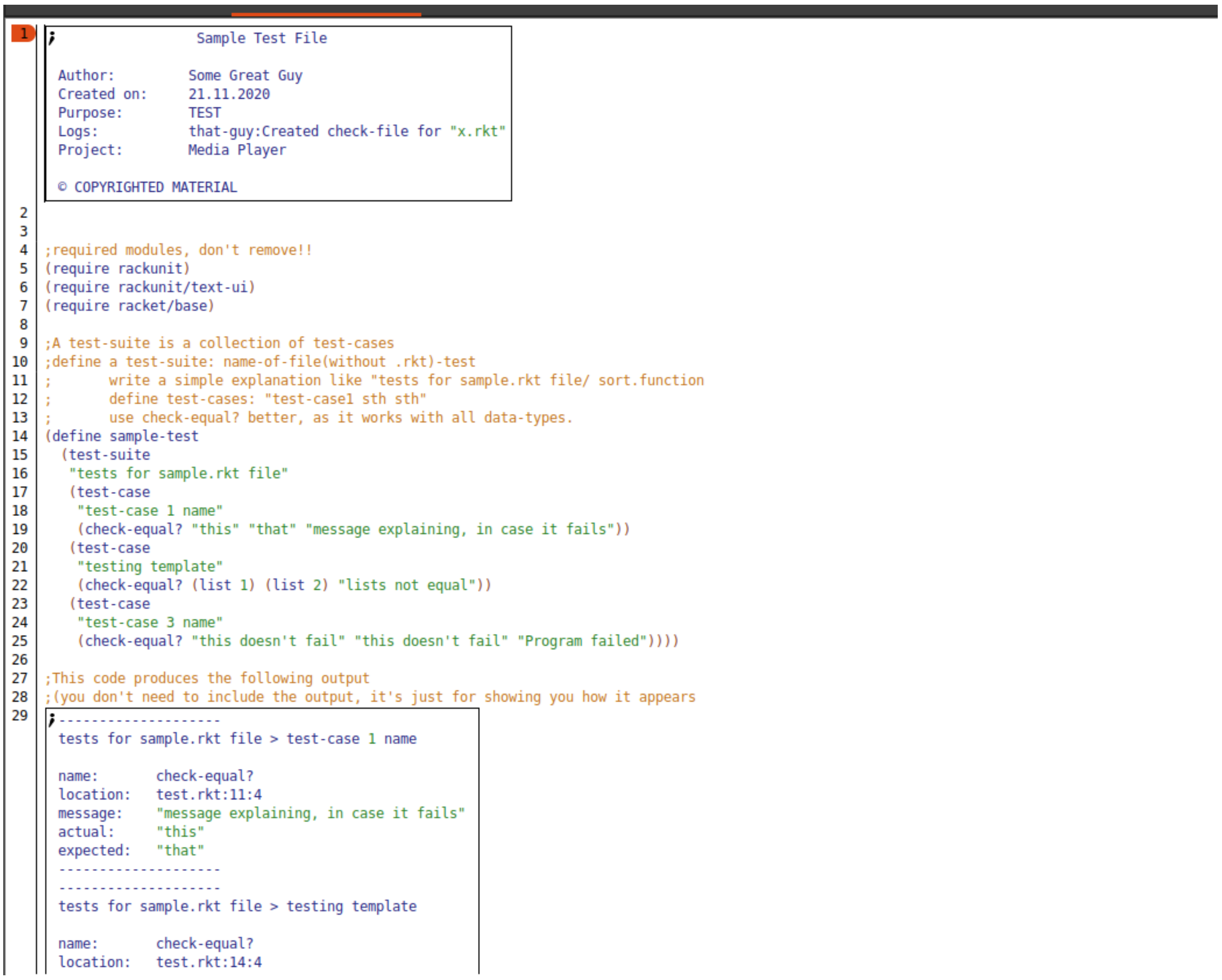
- **write tests along the way**
- tests are written in **separate test-file.rkt** file
- test files go into **test folder**
- write your test cases, in an individual file, provide the check-function (according to the template)
- write the check-function and the require your test file in the **cumulative-test.rkt**

- on the Debugging phase, everybody will review someone else's code,
   and we will perform like registry check for each file, where each file will be
  validated by at least 2 group members.

Templates

A template with a commented header is ready for you to use, please modify it accordingly when you start working.
There are two types of templates:

- A module’s template
- A test template



```
;                   Sample Test File

 Author:         Some Great Guy
 Created on:     21.11.2020
 Purpose:        TEST
 Logs:           that-guy:Created check-file for "x.rkt"
 Project:        Media Player

 © COPYRIGHTED MATERIAL


;required modules, don't remove!!
(require rackunit)
(require rackunit/text-ui)
(require racket/base)

;A test-suite is a collection of test-cases
;define a test-suite: name-of-file(without .rkt)-test
;       write a simple explanation like "tests for sample.rkt file/ sort.function
;       define test-cases: "test-case1 sth sth"
;       use check-equal? better, as it works with all data-types.
(define sample-test
  (test-suite
   "tests for sample.rkt file"
   (test-case
    "test-case 1 name"
    (check-equal? "this" "that" "message explaining, in case it fails"))
   (test-case
    "testing template"
    (check-equal? (list 1) (list 2) "lists not equal"))
   (test-case
    "test-case 3 name"
    (check-equal? "this doesn't fail" "this doesn't fail" "Program failed"))))

;This code produces the following output
;(you don't need to include the output, it's just for showing you how it appears
;--------------------
 tests for sample.rkt file > test-case 1 name

 name:       check-equal?
 location:   test.rkt:11:4
 message:    "message explaining, in case it fails"
 actual:     "this"
 expected:   "that"
 --------------------
 --------------------
 tests for sample.rkt file > testing template

 name:       check-equal?
 location:   test.rkt:14:4
```