UNIVERSITY OF TRENTO
Department of Information Engineering and Computer Science

Tutor me

Course: “Software Engineering”

FINAL DELIVERABLE

By:
Filippo Maffei
filippo.maffei@studenti.unitn.it

Massimo Girardelli
massimo.girardelli@studenti.unitn.it

Tetiana Golovkina
tetiana.golovkina@studenti.unitn.it

Trento – 2023

CONTENTS

PROJECT MAIN OBJECTIVE

The main goal of our tutoring system, TUTOR ME, is to create a user-friendly website that connects students and tutors for personalized one-on-one lessons. Our platform makes it easy for students to find the right tutor, schedule lessons, and communicate effectively. Students can browse tutor availability, book lessons, and manage their reservations hassle-free. We provide a dedicated chat feature for seamless communication between students and tutors, where they can ask questions, share files, and access recorded lessons. To help students make informed decisions, we allow them to leave reviews and ratings for tutors they have worked with. Students have the flexibility to adjust their lesson schedules within 24 hours before the lesson. Tutors are promptly compensated after each completed lesson through our secure payment system. Our objective is to create a reliable and efficient platform that simplifies the tutoring experience for both students and tutors.

GIT STRATEGY

We used GitHub to host the code, the branch the main branch is “main”, commits were hosted there or on temporary branches and then merged, the complete git log can be found here: https://github.com/Massiccio1/swe/commits/main

CI/CD PIPELINE

CI: we are using GitHub actions to perform a series of tests, if all tests are successful the new version is pushed live with a web hook.

CD: the site is hosted on render https://tutor-me.onrender.com/, https://render.com/, on test completion a web hook is called and automatically clones the latest commit and deploys the application live.
An automatic redeployment on every push was in use until tests were implemented.

SECRETS

Secrets values are stored in GitHub secrets and render secrets, for a local instances, a file with template values for the .env file is provided here:
https://github.com/Massiccio1/swe/blob/main/.env%20tempalte.txt

BACKLOG

Volunteers:
- M : Massimo
- F : Filippo
- T : Tanya

Epics:
- Backend
- Frontend

(* 1 effort = ~30 minutes of work)

Table 1 – Sprint backlog

ID: backlog Item	sprint task	volunteer	priority	estimated effort
0: As a user I want to browse the application with an interface	frontend header and overall style	T	60	6
1: as the product owner I want my application to be always updated	1.GitHub CI	M	80	6
	2.render complete CD	M		1
2: test implementation for every category	1.Integration with CI/CD	M	70	4
	2.test code template	M		6
	3.test for authentication	M/F		3
	4.test for student	M/F		5
	5.test for tutor	M/F		3
	6.test for course	M/F		3

	7.test for prenotation	M/F		3
3: swagger documentation adjustement	swagger document and validation	F	80	5
4: As a user/tutor I want to login in the browser	front end login	T	60	4
5: As a user/tutor I want to sign up the browser	front end signup	T	60	4
6: As a student I want to see the available courses	1) frontend script request	M	50	2
	2) HTML paging	M		2
	3) Redirect to course page	M		2
7: As a student i want to make a reservation for a course	1) frontend script request	M	50	2
	2) HTML paging			1
	3) check for collision in slots			1
8: As a student I delete a reservation I made	1) frontend script request	M	40	1
	2) HTML paging			1
	3) check for collision in slots			1

user story ID	Effective effort remaining																		
	substory	28/06	29/06	30/06	01/07	02/07	03/07	04/07	05/07	06/07	07/07	08/07	09/07	10/07	11/07	12/07	13/07	14/07	15/07	16/07
0	—	0																		
1	1)	1	1	0																
	2)	0																		
2	1)	4	4	2	2	2	2	2	2	2	1	1	1	1	0					
	2)	5	3	3	3	3	3	3	3	3	0									
	3)	3	3	3	3	3	3	2	2	2	1	1	1	1	1	0				
	4)	4	3	3	3	3	3	3	3	3	0									
	5)	3	3	3	3	3	2	2	2	2	0									
	6	3	3	3	3	3	3	3	3	3	3	3	3	3	0					
	7	3	3	3	3	3	3	3	3	3	3	3	3	2	1	0				
3	—	5	5	5	5	5	5	5	5	5	5	5	5	4	4	2	2	2	2	0
4	—	0																		
5	—	0																		
6	1)	2	2	2	2	2	2	2	2	2	2	2	2	0						
	2)	2	2	2	2	2	2	2	2	2	2	2	2	0						
	3)	2	2	2	2	2	2	2	2	2	2	2	2	0						
7	1)	2	2	2	2	2	2	2	2	2	2	2	2	0						
	2)	1	1	1	1	1	1	1	1	1	1	1	1	0						

user story ID	Effective effort remaining																			
	sub story	**28/06**	**29/06**	**30/06**	**01/07**	**02/07**	**03/07**	**04/07**	**05/07**	**06/07**	**07/07**	**08/07**	**09/07**	**10/07**	**11/07**	**12/07**	**13/07**	**14/07**	**15/07**	**16/07**
	3)	1	1	1	1	1	1	1	1	1	1	1	1	0						
8	1)	1	1	1	1	1	1	1	1	1	1	1	1	0						
	2)	1	1	1	1	1	1	1	1	1	1	1	1	0						
	3)	1	1	1	1	1	1	1	1	1	1	1	1	0						

REMAINING STORIES IN THE PRODUCT BACKLOG

Table 2 – Remaining story 1

user story/item	sprint tasks	priority	volunteer*	estimated effort	real effort
as a tutor i want to start streaming the lesson					

Table 3 – Remaining story 2

user story/item	sprint tasks	priority	volunteer*	estimated effort	real effort

as an admin, i want to verify the user data and approve tutors					

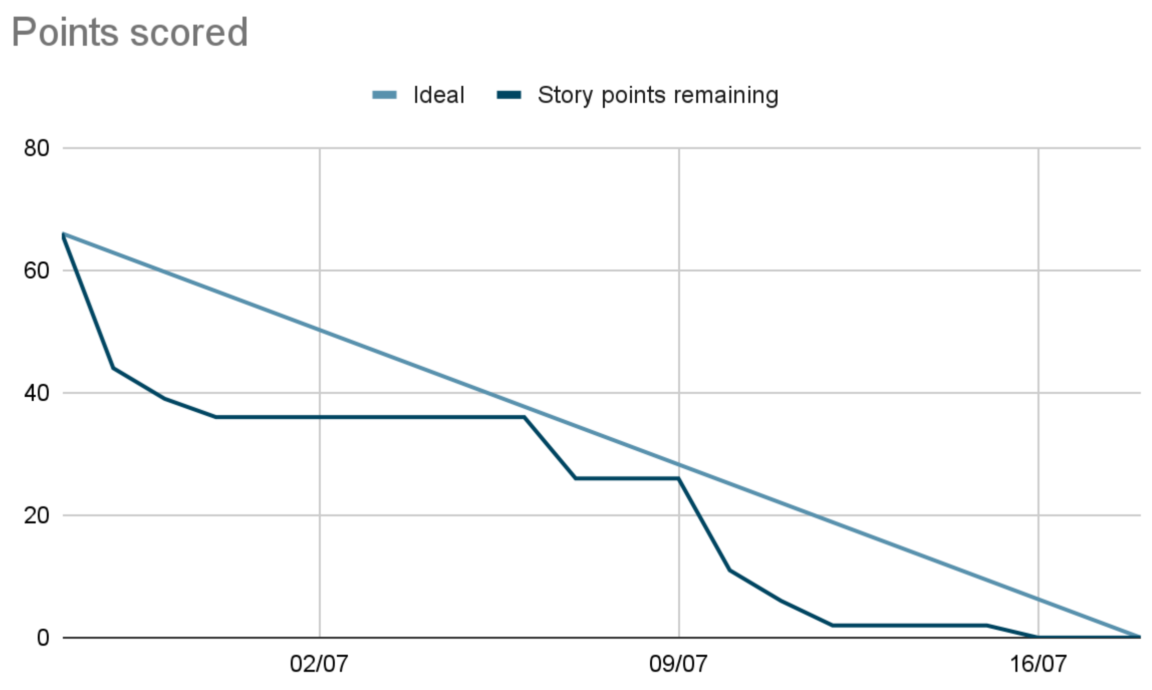
Table 4 – Remaining story 3

user story/item	sprint tasks	priority	volunteer*	estimated effort	real effort
as a product owner I want the student to pay when making a prenotation					

Table 4 – Remaining story 4

user story/item	sprint tasks	priority	volunteer*	estimated effort	real effort
as a product owner I want the new users to verify their account					

BURNDOWN CHART



Picture 1 – Burndown chart with completed work

SPRINT RETROSPECTIVE

What went well:

- Since a good chunk of the API was already completed, we could work on top of the already made requests, for example for the front end.
- More teamwork while fixing bugs in the code.
- Work spread out in different directions, this prevented collisions of commits on the same file.
- More realistic goal for the sprint backlog.
- Effort estimation closer to the real work needed

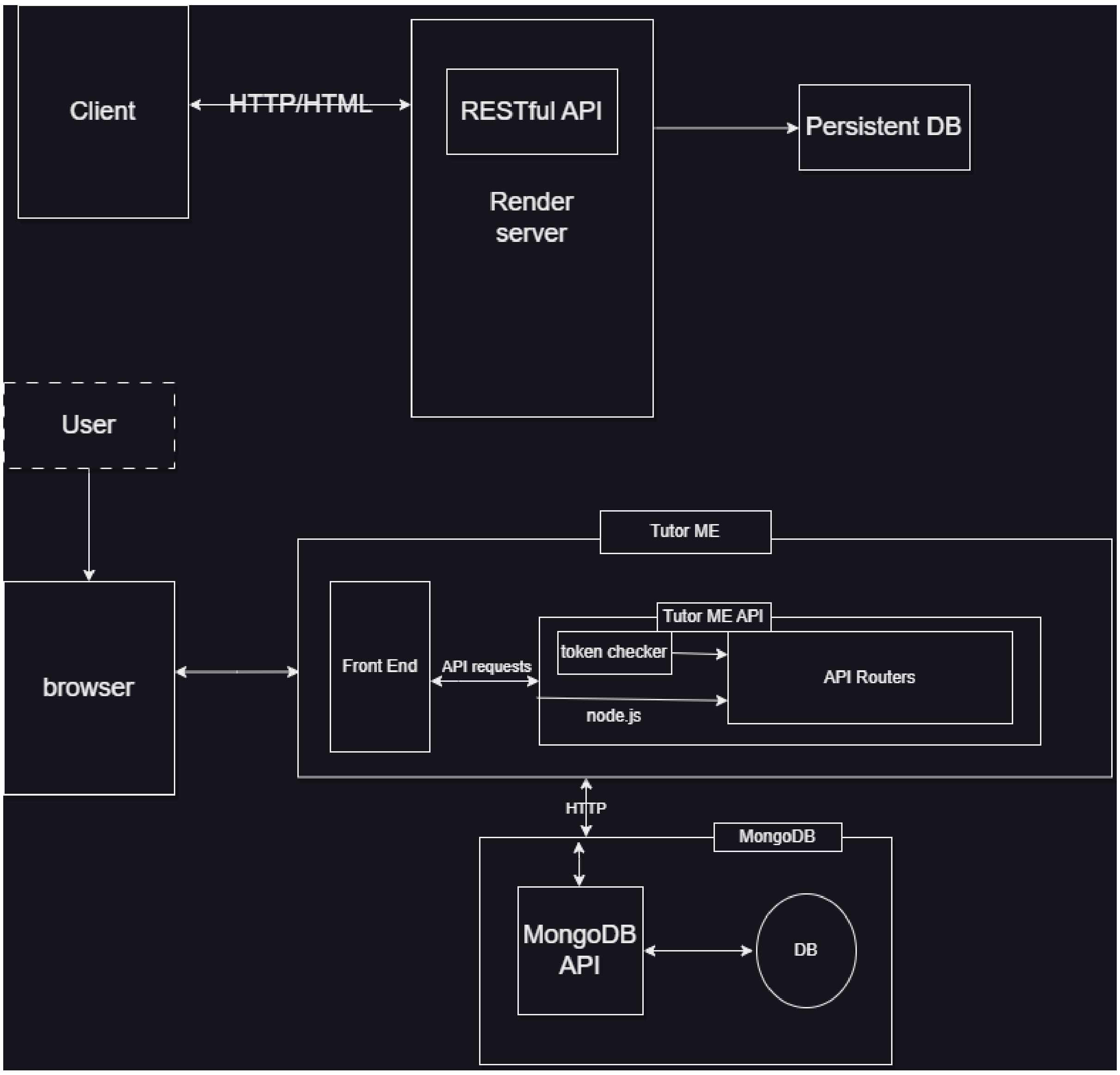
What went bad:

- Due to many exams in June and July, the sprint was spread out and delayed.
- Some items in the backlog require immense work to be completed, unrealistic for this project (ex. streaming lessons).

What we can improve:

— Decide on easier tasks.

ARCHITECTURE



Picture 2 – Architecture definition

TESTING

Table 5 – Test cases

Relative url (from https://tutor-me.onrender.com/api/v1)	METHOD	Description of request	Parameters	Expected output	Test case
/authentications	POST	post student credentials for login	body.email body.password	if credential match, generate a token for the user to use	check for empty fields, missing fields, wrong data types
/authentications_tutor	POST	post tutor credentials for login	body.email body.password	if credential match, generate a token for the user to use	check for empty fields, missing fields, wrong data types
/students	GET	get the list of all students	none	list of all students	check if all students are being retrieved
	POST	posts a new student	body.email body.password	generates a new student in the database	check for empty fields, missing fields, wrong data types and if a student already exists with the same email
/student/me	GET	get personal information	token	prints information about the student and all the prenotations made	check for empty fields, missing fields, wrong data types, and if the token is for a student or a tutor

Continuation of table 4

Relative url (from https://tutor-me.onrender.com/api/v1)	METHOD	Description of request	Parameters	Expected output	Test case
/student/{id}	GET	get email	none	prints the email related to the student id	check that email and id match the student and no other information in sent
/student/ban	POST	bans student	query.token	student should be flagged as banned in the database	check if the token is a token from an admin
/tutors	GET	get the list of all tutors	none	list of all tutors	check if all tutors are being retrieved
	POST	posts a new tutor	body.email body.password body.name body.desc body.slot	generates a new tutor in the database	check for empty fields, missing fields, wrong data types and if a tutor already exists with the same email
/tutors/me	GET	get personal information	query.token	prints information about the tutor and all the prenotations made and the courses available from that tutor	check for empty fields, missing fields, wrong data types, and if the token is for a student or a tutor, also if all courses and prenotation are displayed
/tutors/{id}	GET	get email	none	prints the email, name, description and slots of a tutor	check if data is correct

Continuation of table 5

Relative url (from https://tutor-me.onrender.com/api/v1)	METHOD	Description of request	Parameters	Expected output	Test case
/prenotations	GET	get all prenotations	query.token	gets all the prenotations related to the student of the token	check if only the prenotations of the correct students are show
	POST	post a new prenotation	query.token body.student body.courseId body.tutor body.timeslot	create a new prenotation in the database	check if all the data match a course and the tutor is available in that slot and if the token matches the studentId
/prenotations/{id}	GET	get info about a single prenotation	query.token	get info about a single prenotation	check if the students or tutor has the rights to access it
	DELETE	deletes a prenotation	query.token	removes prenotation	checks permissions and if the prenotation exists
/course	GET	get all courses	none	returns all courses	check if all courses are being retrieved
/course/{id}	GET	gets a single course	none	return 1 course	check if course exists
/course/new	POST	makes new course	query.token body.tutorId body.desc body.price body.Subject	makes a new course in the database related to the tutor	check if token in from a tutor and tutorId all fields are present and datatypes

Continuation of table 5

Relative url (from https://tutor-me.onrender.com/api/v1)	METHOD	Description of request	Parameters	Expected output	Test case
/course/delete/{id}	DELETE	deletes course	query.token	deletes a course	check if token in from a tutor and tutorId matches, check if the course exists

These routes:

- /students/me
- /students/ban
- /tutors/me
- /prenotations
- /course/new
- /course/delete

Pass through a token checker that decrypts the token with a secret key, if the token is valid the account is passed down the other routers so that the id, email and account type can be checked.

HOW TO USE THE WEBSITE

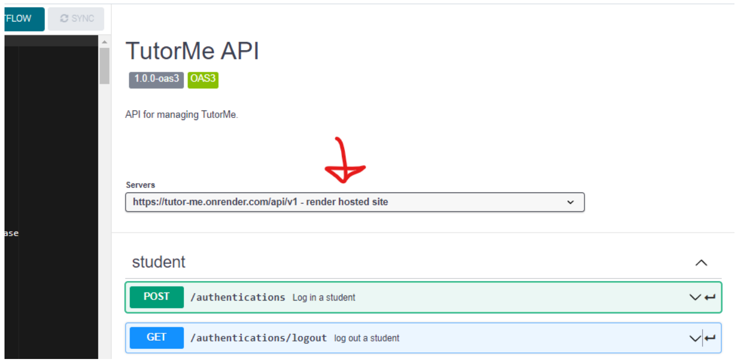
A list of instruction for a demo of the project with the frontend

1) got to https://tutor-me.onrender.com/api/v1/status to wake up the website and the API, a message reporting the server status should appear as soon as the program loads
2) navigate to https://tutor-me.onrender.com/ this is the main page
3) In the top right click on sign up -> I want to be a student -> sign up
4) Use dummy credentials to make an account (credentials might be visible to the developers, use dummy ones)
5) Login in the new account
6) you should be here: https://tutor-me.onrender.com/students/secure/home/ , now decide your field of interest and click on [Find courses] to display all available courses
7) Click on [browse this course] to check if there are available slots
8) if there are you can reserve one otherwise go back and try with another one
9) in the header of the page click on [Bookings] or go to https://tutor-me.onrender.com/students/secure/bookings/ to view your prenotations and delete them
10) go back to the same course and check if the slots are free again

In table 6 there are the credentials for demo users

HOW TO USE THE API

A quick guide on the API endpoints and some way to use them
You can find them on
https://app.swaggerhub.com/apis/FILIPPOMAFFEI2/tutorMe2/1.0.0#/student/post_authentications
All tests are performed by clicking on the endpoint, then [Try it out], then changing the values of the fields and eventually inserting the token and clicking on execute.
IMPORTANT: change the server to the render hosted one to connect with the API



TutorMe API

1.0.0-oas3 OAS3

API for managing TutorMe.

Servers

https://tutor-me.onrender.com/api/v1 - render hosted site

student

POST /authentications Log in a student

GET /authentications/logout log out a student

First of all got to https://tutor-me.onrender.com/api/v1/status or scroll to the bottom to find /status under the tool tag
Not all endpoints will be explained here, just a few
Student:

- POST /authentication is the login and the first step, change email and password fields with the template ones from table 6, in the response body there is a token, copy it to use it later in other requests

- GET /students is to retrieve all students, useful to check when new accounts are made
- GET /students/me to get more in depth information like prenotations and the id found in the self field (only the last number of the path)

Tutor:

- POST /authentication_tutor similar to the student one, just the email and password are required, here also is provided a token that needs to be copied
- GET /tutors/me with token returns also a list of courses held by this tutor
- POST /tutors/me/slot with the tutor token you can post an array of slots, the body should be something like this:
  ```
  {
    "slot": [
      0,1,2,3,4,5
    ]
  }
  ```

Prenotations:

- GET /prenotations , requires a student or tutor token, returns a list of prenotations
- POST /prenotations
- GET /prenotations/{id} retrieves a single prenotation

Course:

- GET /course is the list of all courses
- POST /course/new needs a tutor token, requires the tutor ID for redundancy and to better retrieve it

Debug:

This is a list of tools, they are kept public while the project is developing, can be put behind a token checker at the end of development. if the reset functions are used, they should follow the order student->tutor->courses->prenotations

In the unfortunate case of a crash, the system can be rebooted with the link: bit.ly/43v3GgT followed by database resets in the debug category.

DEMO CREDENTIALS

Table 6 – Demo credentials

Type	Email	Password
Student	e1@gmail.com	p1
	e2@gmail.com	p2
	e3@gmail.com	p3
	e4@gmail.com	p4

Tutor	t1@gmail.com	p1
	t2@gmail.com	p2
	t3@gmail.com	p3
	t4@gmail.com	p4

ADDITIONAL LINKS

1. Site:
   1.1. https://tutor-me.onrender.com
   1.2. (since render doesn't keep the app up 24/7, check the status on https://tutor-me.onrender.com/api/v1/status)
2. Swagger API:
   2.1. https://app.swaggerhub.com/apis/MASSIMOGIRARDELLI/TutorME/1.0.0
   2.2. https://app.swaggerhub.com/apis/FILIPPOMAFFEI2/tutorMe2/1.0.0
3. Repository:
   3.1. https://github.com/Massiccio1/swe
4. crash recovery:
   4.1. bit.ly/43v3GgT