UNIVERSITY OF THESSALY

SCHOOL OF ENGINEERING

DEPARTMENT OF ELECTRICAL AND COMPUTER ENGINEERING

System Dynamics Web Platforms for Educative Purposes

Diploma Thesis

Stelios Lagaras

Supervisor: Aspassia Daskalopulu

September 2022

UNIVERSITY OF THESSALY

SCHOOL OF ENGINEERING

DEPARTMENT OF ELECTRICAL AND COMPUTER ENGINEERING

System Dynamics Web Platforms for Educative Purposes

Diploma Thesis

Stelios Lagaras

Supervisor: Aspassia Daskalopulu

September 2022

ΠΑΝΕΠΙΣΤΗΜΙΟ ΘΕΣΣΑΛΙΑΣ

ΠΟΛΥΤΕΧΝΙΚΗ ΣΧΟΛΗ

ΤΜΗΜΑ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ

Πλατφόρμες Δυναμικών Συστημάτων και Εκπαιδευτικά Εργαλεία

Διπλωματική Εργασία

Λαγάρας Στέλιος

Επιβλέπουσα: Δασκαλοπούλου Ασπασία

Σεπτέμβριος 2022

v

Approved by the Examination Committee:

Supervisor **Aspassia Daskalopulu**
Associate Professor, Department of Electrical and Computer Engineering, University of Thessaly

Member **Yeoryios Stamboulis**
Associate Professor, Collaborating Instructor, Department of Economics, University of Thessaly

Member **Alexander Chroneos**
Professor, Department of Electrical and Computer Engineering, University of Thessaly

Acknowledgements

Στην αγαπητή μου Οικογένεια,
την Ζανί και τους φίλους μου

Θερμές ευχαριστίες στον αναπληρωτή καθηγητή κ.Σταμπουλή Γεώργιο,
για την καθοδήγηση και το κίνητρο που μου έδωσε, καθώς και στην
αναπληρώτρια καθηγήτρια κ.Δασκαλοπούλου Ασπασία για τις διορθώσεις.

DISCLAIMER ON ACADEMIC ETHICS
AND INTELLECTUAL PROPERTY RIGHTS

«Being fully aware of the implications of copyright laws, I expressly state that this diploma thesis, as well as the electronic files and source codes developed or modified in the course of this thesis, are solely the product of my personal work and do not infringe any rights of intellectual property, personality and personal data of third parties, do not contain work / contributions of third parties for which the permission of the authors / beneficiaries is required and are not a product of partial or complete plagiarism, while the sources used are limited to the bibliographic references only and meet the rules of scientific citing. The points where I have used ideas, text, files and / or sources of other authors are clearly mentioned in the text with the appropriate citation and the relevant complete reference is included in the bibliographic references section. I also declare that the results of the work have not been used to obtain another degree. I fully, individually and personally undertake all legal and administrative consequences that may arise in the event that it is proven, in the course of time, that this thesis or part of it does not belong to me because it is a product of plagiarism».

The declarant

Stelios Lagaras

Diploma Thesis

System Dynamics Web Platforms for Educative Purposes

Stelios Lagaras

Abstract

Simulations are widely accepted as a modern tool in e-learning and education, due to the fact of representing complex processes and phenomena in a quality and engaging manner. Today, the various simulation software are costly and hard to be used by tutors and students. For this reason, we develop a simulation learning platform with modern tools, where users can easily run various models via a friendly interface, tutors can parameterize the exposed variables of the model, while all the gathered data are stored appropriately in a database for further analysis.

Keywords:

System Dynamics, Learning Simulations, Learning analytics, e-learning, Vue framework, Heroku, FastAPI, Python, Javascript, Vensim

Διπλωματική Εργασία

Πλατφόρμες Δυναμικών Συστημάτων και Εκπαιδευτικά Εργαλεία

Λαγάρας Στέλιος

Περίληψη

Η προσομοίωση αποτελεί σύγχρονο αναγνωρισμένο εργαλείο για την εκπαίδευση με ψηφιακά μέσα, καθώς δίνει την δυνατότητα στον εκπαιδευόμενο να αντιληφθεί εικονικά και να αλληλεπιδράσει με το αντικείμενο εκμάθησης. Σήμερα, τα διάφορα λογισμικά προσομοίωσης αποτελούν εξεζητημένο και κοστοβόρο υλικό το οποίο είναι δύσκολο να χρησιμοποιηθεί άμεσα από εκπαιδευόμενους και εκπαιδευτές. Για τον λόγο αυτό, αναπτύσσουμε μια πλατφόρμα εκμάθησης μέσω προσομοιώσεων όπου με την χρήση νέων εργαλείων οι χρήστες μπορούν εύκολα να διατρέχουν διάφορα μοντέλα μέσα από φιλικό περιβάλλον, οι διδάσκοντες μπορούν να τα παραμετροποιήσουν κατάλληλα, ενώ όλα τα δεδομένα αποθηκεύονται σε μια βάση δεδομένων για την περαιτέρω ανάλυση και επεξεργασία τους.

Λέξεις-κλειδιά:

Συστημική Δυναμική, Εκπαιδευτική Προσομοίωση, Ανάλυση δεδομένων μάθησης, Ηλεκτρονική μάθηση

Table of contents

List of figures

List of tables

Abbreviations

SD	System Dynamics
js	Javascript
LTG	Limits to Growth
DaaS	Database as a Service
ORM	object relational mapping
UI	user interface
SPA	Single Page Application
auth	Authentication
LMS	Learning Management Systems
LA	Learning Analytics

Chapter 1

Introduction

1.1 Aim of the Dissertation

The aim of this dissertation is the creation of a full stack web platform, which serves as an interactive learning environment running simulation models in System Dynamics. The creation of a friendly interface will help students to create knowledge from running simulations, while they inspect various models parameters and experi- ment with them. Our platform PySims, saves the progression of the simulation the parameters values and the final results, enabling tutors to observe and potentially analyze students approaches on various simulated situations (problem conditions) with the aim to evaluate their way of thinking and their behavior and to give personalized feedback back to them.

1.2 Justification of the research topic

The rapid technological improvement and integration in everyday life has affected the educational procedure with the usage of internet telecommunications and computer devices, leading to a wide learning ecosystem [1.1] which is called E-Learning. E-Learning has been around since the 1960s, but it is only in the last two decades that there is a tremendous growth in new applications of e-learning due to the rapid developments in technology and computers, like smartphones, virtual reality systems, IoT (Internet of Things), computational power etc. On

The first stage, E-Learning, has been focused on managing, storing and serving information via LMS (Learning Management Systems), while the evaluation of acquired knowledge is happening through questionnaires forms, thus, leading to a very static and passive experience on students' part. The revolution of Web 2.0, has invented new sub-fields in E-Learning like Microlearning [4], where the information is broken down to smaller perspectives, Mobile Learning, with the invention of smartphones and tablets, where learning can exists in any location, Open Education which focus on the accessibility of learning material, freely for everyone, Do-it-Yourself Learning, which provides just in time, just enough information and guidance for the users. All the above, shapes into E- Learning 2.0 [1] plus the new technological trends, like virtual & augmented reality may lead to future extensions and research interest.

As a more dynamic environment emerges, simulations are suitable with various course subjects as a tool in the e-learning procedure.Various researchers propose that. ”well-designed simulations will develop in the student a profound, flexible, spontaneous kinesthetic understanding of the subject matter” [5] [6] [7]

Simulation software has been developed in a classical architecture of applications since the mid 20th century. Although numerous of these softwares offer a standalone toolbox for analysts, there are lacking in online connection, like modern web apps, and in holistic ways of group management and learning capabilities.

In the Big Data era, (learning) analytics are constantly increasing of interest, various stakeholders like compa- nies, corporations, state departments, educational institutions and research departments are looking for data tanks, in order to understand various behavioral patterns and strategies.

In our case, we are looking forward to interconnecting the various puzzle parts of System Dynamics and Simulations, creating a modern, compact, and reliable solution for collecting the necessary data.

1.3 Structure of the dissertation

In Chapter 2 we present more details about e-learning, how simulation tools are used in e-learning, what are the benefits for the learners and the tutors, and what kind of pitfalls should be avoided. In Chapter 3 we discuss the architecture of our application. PySD[8] is our core library responsible for the simulation execution, FastAPI will be used as the API framework for building the functionalities that our application should accomplish, like

storing and serving simulations, writing and reading results from and on the database. On the front end part of our platform we choose Vue 3 and Element plus UI library for designing user's interface and communicating with the back end. In Chapter 4, we present more technical details and some typical user scenarios. We conclude with future ideas of using these tools in the classroom describing a road map for our next plans.

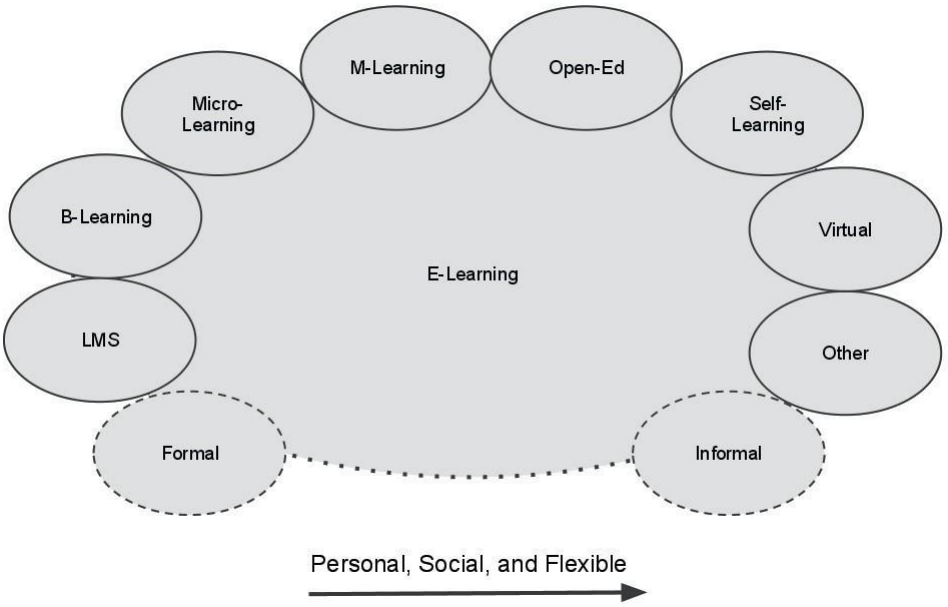

Figure 1.1: E-Learning Ecosystem [1]

Chapter 2

Simulation as a learning Tool

2.1 Types of E-Learning Tools

The variety of E-Learning tools are constantly developed in parallel with technological tools and devices. From the classical blogs and wikipedia articles we are facing an expansion to podcasts and social media usage for learning purposes. Virtual worlds tend to add interactivity and familiarity to the learners, with avatars and interactive 3D environments [9] as also augmented and virtual reality are coming to enhance the users experience. Open source repositories, open learning communities and collectives are building cutting edge portals like stack overflow, and paid platforms for developers like codemy, codecademy etc.

2.2 Terminology of Simulation, Game, Simulation-Game

Before we present an historical overview of simulations, allow us to clarify the meaning and overlapping of the words Simulation, Game and Simulation-Game. Games can exist in a pure entertainment sphere, for example running or playing rock-paper-scissors, that kind of games are also usually called plays. On the other hand, simulations are not always games, for instance a weather forecast simulation, or a virtual representation of a model like Holzinger et al. (2009)[10], where they present a medical education simulation. We can distinguish at least two cases where a simulation-game coexists. First, in many known games which are fundamentally bases in some

kind of a simulation, e.g. Flight Simulator, The Sims, Age of Empires etc. simulations are used in the background with play centric target of the game, where in the second case, we found simulation centric scenarios, like the famous Beer Game, developed at the MIT (Massachusetts Institute of Technology) in the 1960s [11] where the term game is mostly used to describe the educational, training purpose of the simulation - we also find the term **serious games** for such cases.

A very interesting article about John Hiles, a simulation pioneer and his company, Maxis Business Simulations[3], points out why games and simulations are communicating vessels. Maxis and their circle, for a decade, were jumping between the corporation specific simulations, and simulation games. It is in our special interest to study the intersection of these two worlds, where simulation meets the game and vice versa, such, there we found most of the learning and training potential.

2.3 Modern implementation of Simulation Games

Pasin & Giroux has made a very good overview of modern implementation of simulations in their article[12]. Some of the remarkable modern cases are the flying simulators where pilots have real life training experience, to virtual reality systems teaching surgical techniques to students, simulations are helping students to take a virtual real hand on experience in difficult tasks. Another common field of application is policy making decision problems, and crisis management [13]. This connects to Urban simulations and aspects like land-field, water and energy consumption policies [14] ,and of course the health sector on various pandemic scenarios, urban pollution ,urban mobility, and habitual health metrics like smoking and safe driving. One of the first impactful books in SD was Limit to Growth(1972)[15] by Dennis Meadows. His vision about the limitation of our planet's resources in an era that the industrialization was dominating was remarkable. Last but not least, the management sector, has appreciated earlier on 1950s the value of simulations as a training tool. According to Pasin and Giroux [12], by 1961 "more than 100 business games had been published in the US alone..." [16], and by 1998 more than 60% of companies with more than 500 employees were using simulations games in their training activities according to Faria's survey[17]. The trending on management training keeps on increasing in the 20s as now it is applied in specific management fields like, project management, knowledge management, financial management etc.[12].

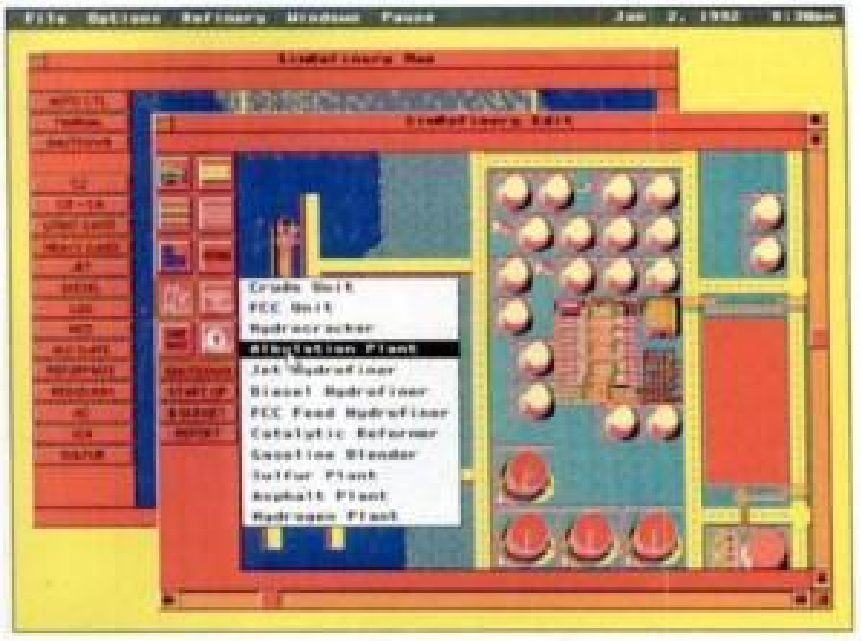

Figure 2.1: One of the few surviving screenshots of SimRefinery.[2][3]

2.4 Benefits of simulation based learning

Simulation as a learning tool can benefit and extend the classical learning procedure. Simulation can be useful by revealing the interconnections of a System, and how small actions of seeming independent parts apparently affects each other, thus revealing what is called the big picture. Pasin, writes down ”They allow participants to develop a global perspective, to connect learning with real-world situations and to get close to the realities...” [12][18]. Another development that we should consider, is technological improvements and how the new generations are familiarized with information interaction via gaming and everyday computer usage. That, as stated from Proserpio and and Gioia [19] applies in the modern needs of ”virtual generation” (V-gen) which is much more visual, interactive, and focused”. Some of the positive attitudes recorded from Papoutsakis et.al [20], where a greek teachers study group examined, are: the teachers found simulations as a pleasant way of teaching, the students are interested in being actively involved, they are modern and give plenty of options to the tutors. Stergioulas[21] points out that simulation tools enhance the student centered learning orientation, allowing students to practice decision making and critical thinking and to stage the consequences of their applied strategies. All the

above, makes simulation tools as a necessary addition in the 21st century classroom.

2.5 Challenges of simulation representation

Although simulations can be a very useful tool, they need to be designed with precision and awareness of learners' perception, as, very easily, can be turned into an overwhelming experience for them. Van der Meij et al. [22] records four tasks that learners have to cope with. First they must get familiar with the format and the operators of the simulation. Various attributes like labels, axis and variable names should be placed in a qualitative manner. In summary of the rest, the represented domains have to be cleared, learners could relate the shared information between graphs, and the translation between them should be easy. Potential problems that may occur for the learners, are over-generating cognitive load, thus minimizing the resources for actual learning [23]. Also, the ability to connect visual representation with verbal and written information was reported to show signs of expertise by at least two studies from Tabachneck et al[24], and Kozma [25]. Our simulation interface has been carefully designed and implemented to make a pleasant experience for the learners, while keeping the necessary information on the layout.

2.6 Collecting Data and Learning Analytics

Learning Analytics concept is another constantly growing dimension of E-Learning as it examines and shapes the procedure of learning, the way that learning is implemented and continuously creating a loop by gathering information from learners, their actions and their results. The definition given by the Society of Learning Analytics Research is: “Learning analytics is the measurement, collection, analysis and reporting of data about learners and their contexts, for purposes of understanding and optimizing learning and the environments in which it occurs” [26]. Siemens points out that corporation private usage of LA from big techs like Facebook, Netflix and Shazam are ahead from establishing defined techniques and tools, although that points out the necessity that arises. One of the functionalities implemented in our platform is storing all the simulation progression and the decisions taken by the learners, so future research and analysis can be done by using LA techniques. In further steps, we may analyze

how learners take their decisions, how successfully they adapt and if our learning model is successfully agile.

Chapter 3

Application Architecture

3.1 Software review and motivation for PySD

From the later 90s there have been developed a lot of Simulation Specific Software systems some of them worth mentioning:

- Powersims
- Vensim
- Stella & iThink
- Anylogic
- Insight Maker

It is highly interesting that so many software products share the pie of specific usage of them, the bigger clients are corporations from fields like Oil Companies, Railway Companies, Real Estate and other high value aspects. But, as it happens with many high-tech products of our era, **compatibility** is a real problem. While every company has its own file encryption it is impossible to iterate between them with the same model-file. There have been a try for a unified file type XMILE [27] which interpolates with Stella & iThink, and also it is on the roadmap of our core tool PySD[28], but till then, we will focus on Vensim files which will be parsed/translated to python files.

3.2 Application architecture - back end

3.2.1 Simulation core engine

PySD is a novel approach on Simulating SD models by means of bringing the necessary tools in one layer to the bottom of software development, just into a python package. As the creators state ”System dynamicists should directly use the tools that other people are building, instead of replicating their functionality in SD specific software”. Now, the developer is able to have a pandas dataframe with all the results with a few lines of code. Also, PySD offers the necessary functionalities that our application needs, like running a simulation from a previous state and changing the parameters of the model variables.

3.2.2 FastAPI back end

It is a common practice on modern applications decoupling the back end with front end by creating an API which serves as the intermediate traffic regulator. Our API will get simulation calls, process them, run the simulation and return the results on the front end, thus, it is responsible for a variety of tasks and its role is of high importance. As a mechanical analogy API connects our engine (PySD) with the wheels and the chassis (front end) of our vehicle (application).

As PySD is developed in python language, naturally we choose to develop our API in python so our next task is to choose one of the modern Python API frameworks. Our choice is FastAPI, an open source project from Tiangolo which is notorious for its speed and also has a great documentation.

3.2.3 Database

We use `https://dbdiagram.io/d` to draw a database sketch prototype

Our main table is *simulation*, which represents the necessary details of a simulation run. Also, we may need another table in order to store the models and their details, such as their namespace -information about each model’s component - and their documentation. Lastly, we may of course have a database table about the user

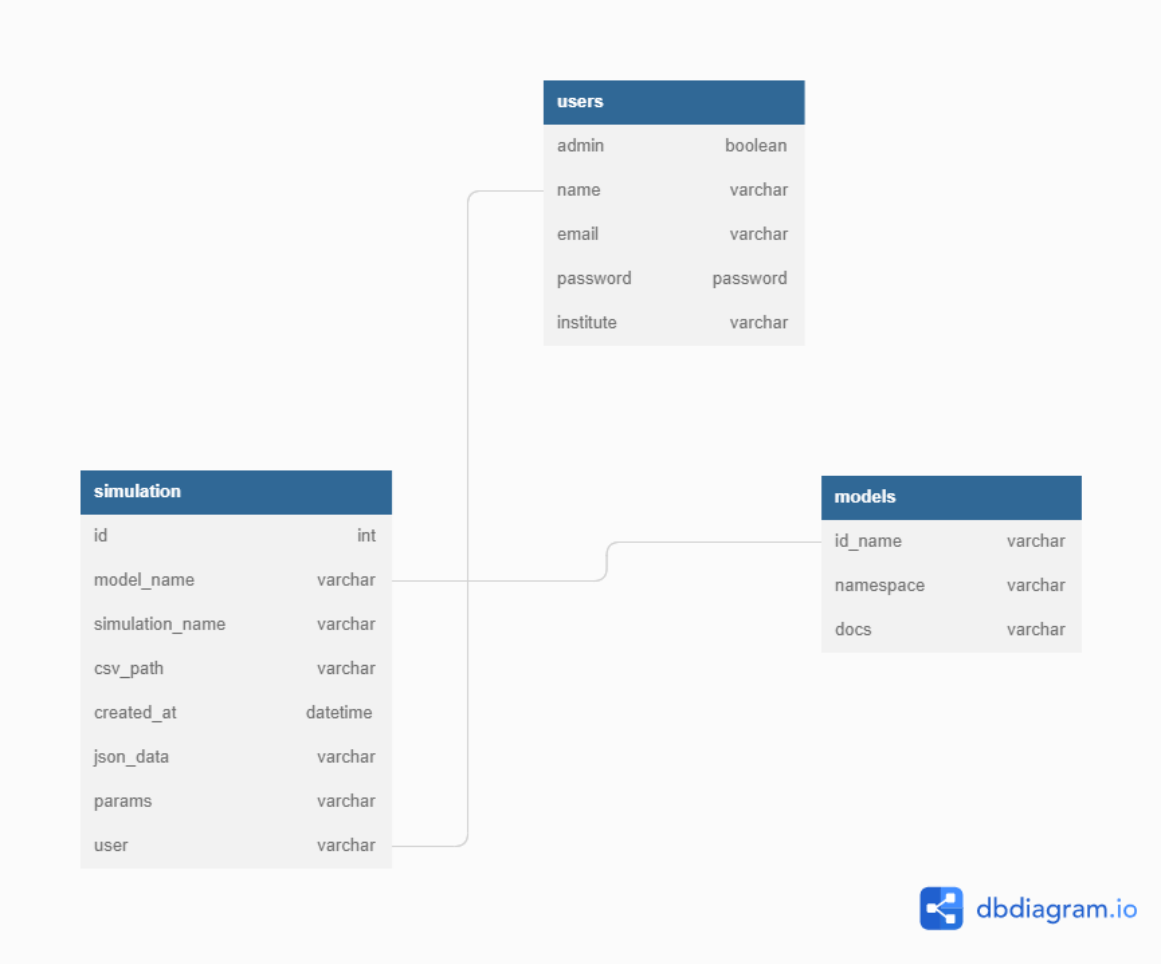

Figure 3.1: Prototype of ER Database

administration and information. In a lot of modern projects, Database as a Service (DaaS) solutions are preferred, helping the developer focus on other crucial parts.

ORM MODEL and SQLAlchemy

The original language to communicate with a Relational Database, has its own syntax, usually the commands- queries are in capitals and some mathematical expressions like joins are also used. In contrast, in the world of object oriented languages, object-relational-mapping (ORM) has been developed to abstract the database from low-level manipulation. With an ORM, you normally create a class that represents a table in a SQL database, each attribute

of the class represents a column, with a name and a type. Then, we can access our database items like a normal Object. SQLAlchemy is the modern tool in python for this job and it's database agnostic, we can focus on development and if we want to switch to another database, we just have to change one line of code and SQLAlchemy's engine will do the work for us. At the moment, supported databases are: SQLite, Postgresql, MySQL, Oracle, MS-SQL, Firebird, Sybase and others, most of which support multiple DBAPIs.

3.2.4 SCHEMA and PYDANTIC VALIDATION

Schema, a greek word, describes the structure of any of the traffic handled by the API. We define different schemas for get requests and responses, thus getting a fixed description of the json structure and also the various field types. Schemas are used on the backbone of FastAPI to validate a request and its response by its schema and the field types, thus, improving development ergonomics, minimizing errors and helping with debugging them. Another feature of schemas auto-generates our's FastAPI's documentation tool named "Swagger UI" which we can access in the same url, e.g. "API_URL/docs". There, we can try out our API interactively, performing all of the four actions: post, get, delete, edit. The automatic documentation is particularly useful for other applications and their teams, which may interest using our API. When they would like to use a functionality, e.g. get some data from the database they only need to find the correct function from the documentation and call it, that separation of concerns makes API a black box for external colleagues.

3.2.5 Example of fastapi function

Now we are ready to create our main paths and their functionality, let's take for example we want to define:

```
/get_simul_by_id/{id}
```

on FastAPI would write:

```
@app.get('/get_simul_by_id/{id}', response_model= schema.Simulation)
def get_simul_by_id(id:int, db: Session = Depends(get_db)):
    # DB query to get simulation
```

```
...
return(obj)
```

The first line defines the path where a request hit's, and the at '@' character which is called a decorator shows that this call will be handled by the function defined below. Also we can spot two other syntax patterns, first the brackets '' defines query parameters which can be used from the handler function and also a db: Session call which represents the connection between the API and the DB.

FastAPI also uses a number of helpful tools which integrate with the database and type safe definition which also helps in editor tools, for example in VS code Intellisense tool. We have quick look on them bellow:

Figure 3.2: Tools used to build the API

3.3 Application architecture - front end

Javascript Frameworks have been a recent progression in front end development. The complexity and scale of modern projects demands a structured way of development, reli- ability and reusability. In the last decade there have been dozens of frameworks created, some of them more opinionated like Angular, and some other more flexible like React. In the middle of them, our choice, the Vue framework was invented by Evan You in 2014.

The fact that it is one of the most beloved ones[29], it is a progressive framework and has an active community led as on that choice. Nevertheless, because of the recent change of the default version, the new 3rd version or just Vue 3 (February, 2022), things are progressing and changing quite quickly, and someone may need a bit of time to adapt at this fast pace. One of the benefits of the new features, the Composition API lets developers write more compact and cleaner code. Also, typescript, a superset of javascript which adds static typing is also supported, and has become the main default for a lot of developers. Another useful tool is vite, ”a build tool that aims to provide a faster and leaner development experience for modern web projects” [30] enables the developer seeing immediately the changes on the DOM while coding in their local environment. Vite has also been invented by Evan You. Last but not least, Nuxt, a meta-framework based on Vue 3 is also on its 3rd release Nuxt 3 and is awaited with enthusiasm from the Vue developers community.

3.3.1 Vue App Structure

Vue is a Component based framework, and the proposed way of using it is through SFCs (Single File Component). SFCs combine the famous trio of HTML, JS and CSS languages in one single file. As it stated in the documentation this strategy does not contrast with separation of concerns rule, as the three parts are inherently coupled while offering a lot of advantages in development ergonomics and codebase maintenance. The writer, having used the later strategy of SFCs, totally agrees with the Vue approach.

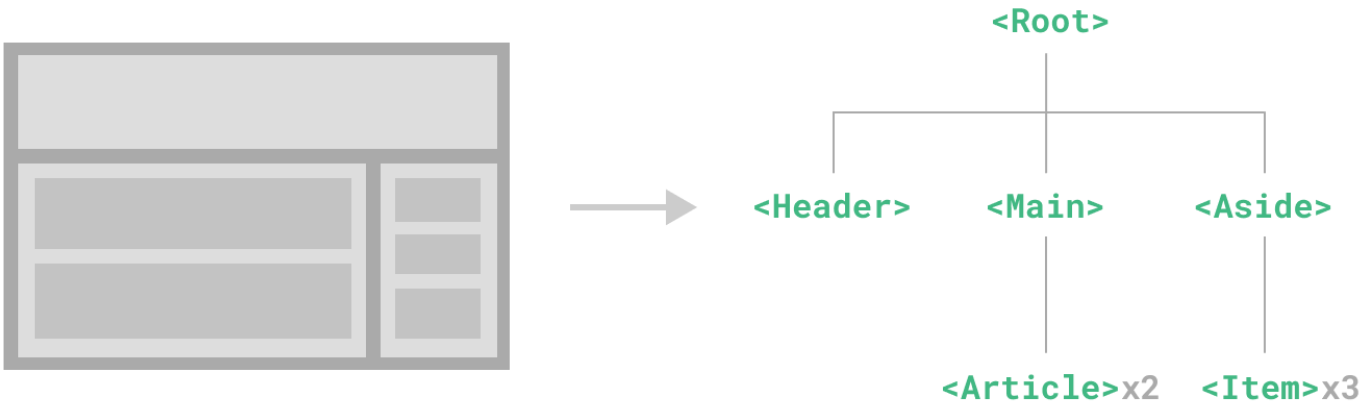

Figure 3.3: Vue Components Structure

3.3.2 UI Frameworks

As frameworks continuously develop, UI component libraries are sticking together. By using a specific UI library instead of writing or mixing different small components, we win a lot of time on styling and micro-development, we get integrity and consistency in our components and a nice aesthetic UI. Three of the UI libraries that have earned a reputation in the community are 1) Vuetify, 2) Qasar and 3) Element+ UI. From them, the two lasts are available on Vue 3, and a lot of people are waiting for the official Vuetify 3 release, which is in a beta state for the time being. Our choice is Element+ UI, which offers a tone of components, is well documented and has extra features like theming and dark mode. All the components from Element+ has the prefix `el-{name of the component}`

Chapter 4

Technical Analysis

4.1 Setting the environment

To begin with, we need to set up our virtual environment for the back end. As we can read in python's official docs "Python applications will often use packages and modules that don't come as part of the standard library. Applications will sometimes need a specific version of a library, because the application may require that a particular bug has been fixed or the application may be written using an obsolete version of the library's interface.The solution for this problem is to create a virtual environment, a self-contained directory tree that contains a Python installation for a particular version of Python, plus a number of additional packages." We found this idea also in the deployment stage when we use Docker containers.

4.2 Structure of the directory

We categorize directory files structure as follows:

- Components - Components are reusable SFCs which can be instantiated on multiple views
- Views - It is the equivalent of a page in a SPA, every page usually has multiple components as children 19

- App.vue - The head of the Vue App. Everything hang's from this component. Usually a router component exists which is responsible for exchanging between the multiple views.

- Composables - It is the equivalent of a local library, where we store functions that are used multiple times by importing them from composables.

- Linter and Prettier - A linter helps by checking that quality coding rules are followed, specifically for vue 3 there is the corresponding eslintrc with the appropriate configuration like ts support, vue rules level of stringency etc. On the other hand, Prettier help's for aesthetic typing rules like auto configure spaces,commas and lines width. We just have to save the file and Prettier makes what its name stands for.

- Config files - everything that has to do with the configuration of typescript support and vite bundler e.g. localhost port, vue experimental features etc.

4.3 Steps of a Simulation

4.3.1 Homepage

Table.vue loads all the user's history results which are stored on the database, it hits the `/get_simuls` endpoint and fetches back all the fields of the table `simulations` [”id”, ”model name”, ”simulation name”, ”csv path”, ”date”]| plus, we add the delete row choice which also delete the instance on the database via `/delete_simul_b`

The `key_id` part of the path, is a path parameter and can be read from FastAPI. We usually use path parameters when we have a small and unique query parameter related with the url's meaning, e.g. here id parameter.

4.3.2 Choosing the Model

We start by describing the functionality of the Tutors control panel. There, the tutor will choose the model for reproduction and open up the constant variables which will be exposed to the students. Firstly, we have initialized/uploaded some available models for testing. This happens by `init-db.py` script as it was described in the

Backend section. Next, we add a dropdown component named el-dropdown-menu and the items with the component el-dropdown-item. The endpoint `/get_available_models` is used to fetch all the available models from the database. The diagram 4.1 bellow, points the visual representation of a fetch request:

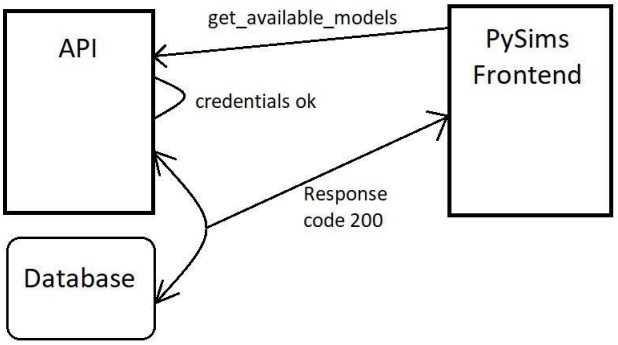

Figure 4.1: API - Frontend connection Sketch

The fetch request is triggered while the `TutorSelection.vue` component is rendering, an improvement technique on this is offered from vue-router, with which we can pre-fetched the data before the component is rendered. More on this topic are mentioned in Vue-router docs and are related with lifecycle hooks of components, see vue-docs on this topic.

4.3.3 Vue state and Pinia

Vue by default stores fetched information in component's scope, and passing properties and emits are used to share information between them. As follows this can be hard for long sequences of children or siblings components route, thus a general state store is used for more complex applications which is accessible from all of the components. The default state plug-in used to be Vuex but the newer Pinia store which abstracts some processes, has become officially the default state package. We also adopt the creation of a store-state, and save there all the necessary information.

4.3.4 DocTable

The `Doctable.vue` component is responsible for tabulating the documentation of the model, offered from PySD library. Every piece of information that hides there, is useful in some part of the App, e.g. type of component as we distinguish the const variables which are exposed to the user, min-max values which should bound the given values of a variable etc. We can see the complete result in figure 4.2

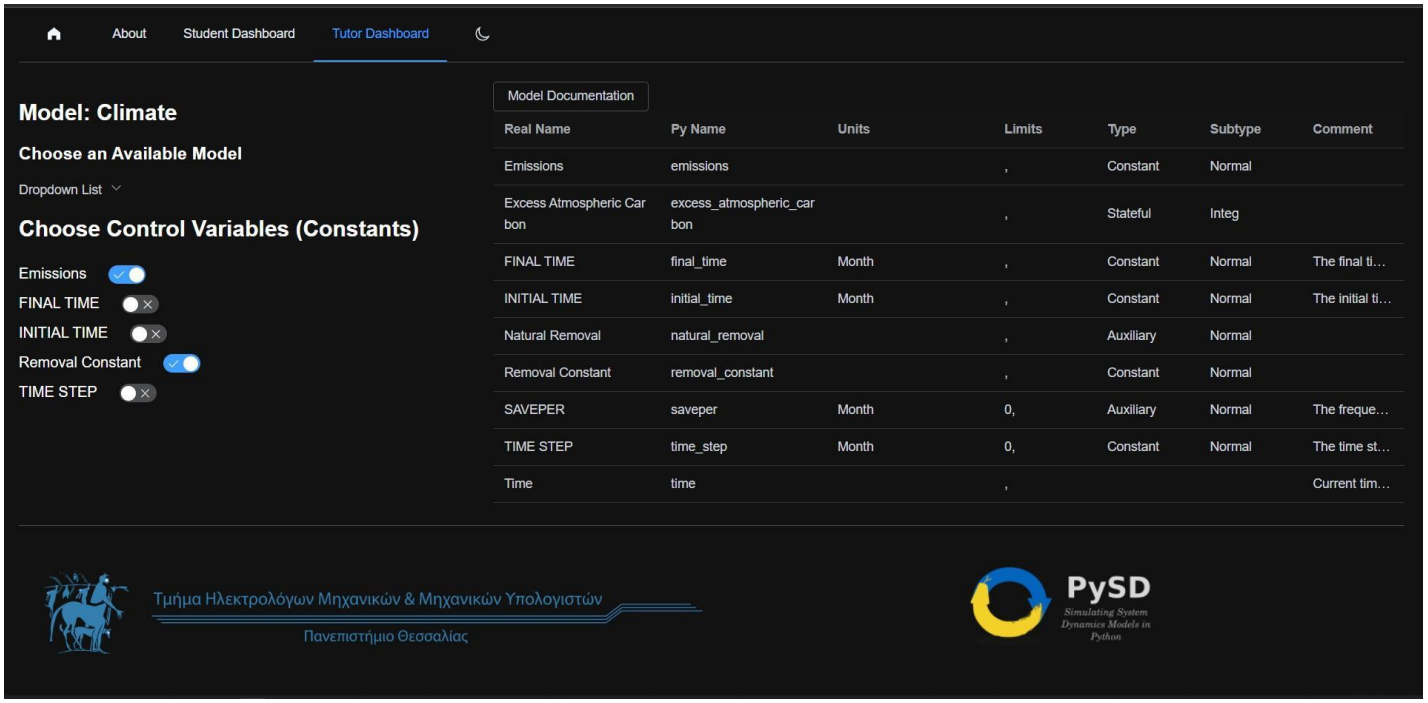

Figure 4.2: Tutor Dashboard

4.4 Student Dashboard

4.4.1 Front-end Side

The Student's Dashboard was one of the heavyweight parts of our project. This needs to handle a lot of parameters in a friendly and intuitive layout. On the right hand of the page stands all the variables information and the progression toolbar with the *run* and *step run* simulation choices. The *step run* process is a much more

complicated one in comparison with the full run, where we just exchange one request with the API via the network. The special boolean path parameter *step_run* is used to inform the API on how to handle the request. So, for example a full run will hit the endpoint: */add_new_simulation/?step_run=false*. We will get back on the API later on to see how that works out. To continue with, we use the parameters that are enabled by the tutor and our app creates a list with sliders (el-slider component) which binds the value to our state *components._value* field. Also, *a special simulation.params* object is created to pass only the students active parameters to the API.We see the right-side part on figure 4.3

4.4.2 API Side

On the `crud.py` folder we have all the necessary functions, where *run-simul* is responsible for running the simulation using the *step_run* parameter to change mode. When the *step_run* is on, the PySD engine runs the simulation for a timestep period and merges the results (a python dictionary) with the latest run. Every *step_run* also loads the latest state from a special .pic file. This is available through the parameter *initial_condition* in *run* function. You can read on the appendix of chapter 1.

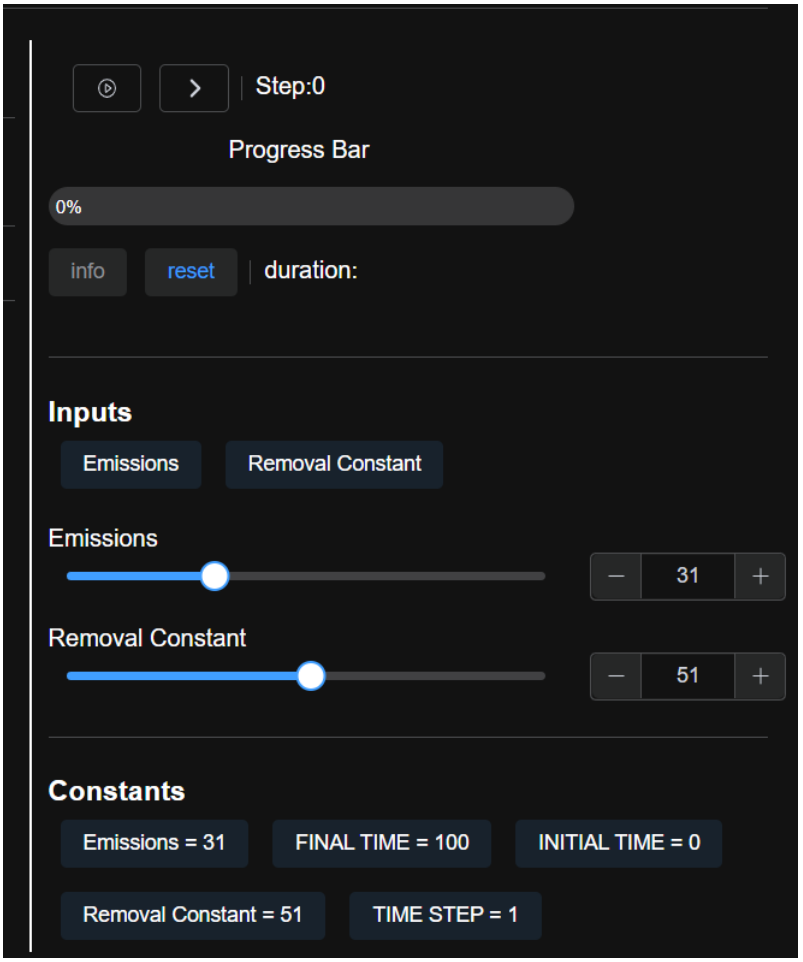

Figure 4.3: Controls of User

chart-js graphs

Chart.js is the responsible tool for rendering our simulation results. We have created our *ChartSimul.vue* com- ponent where we pass the simulation results as a property (Vue prop) as also a *chartid* property which is binded in canvas id tag, in order to make multiple charts which should have a unique id. To make every graph color unique, we map every variable's name (string) to a hex color. In a future version, the user or admin could choose which variables would be rendered together in the same graph. The healthy structure of the *ChartSimul.vue* component, allows us to populate our desirable graph and render it scaled.

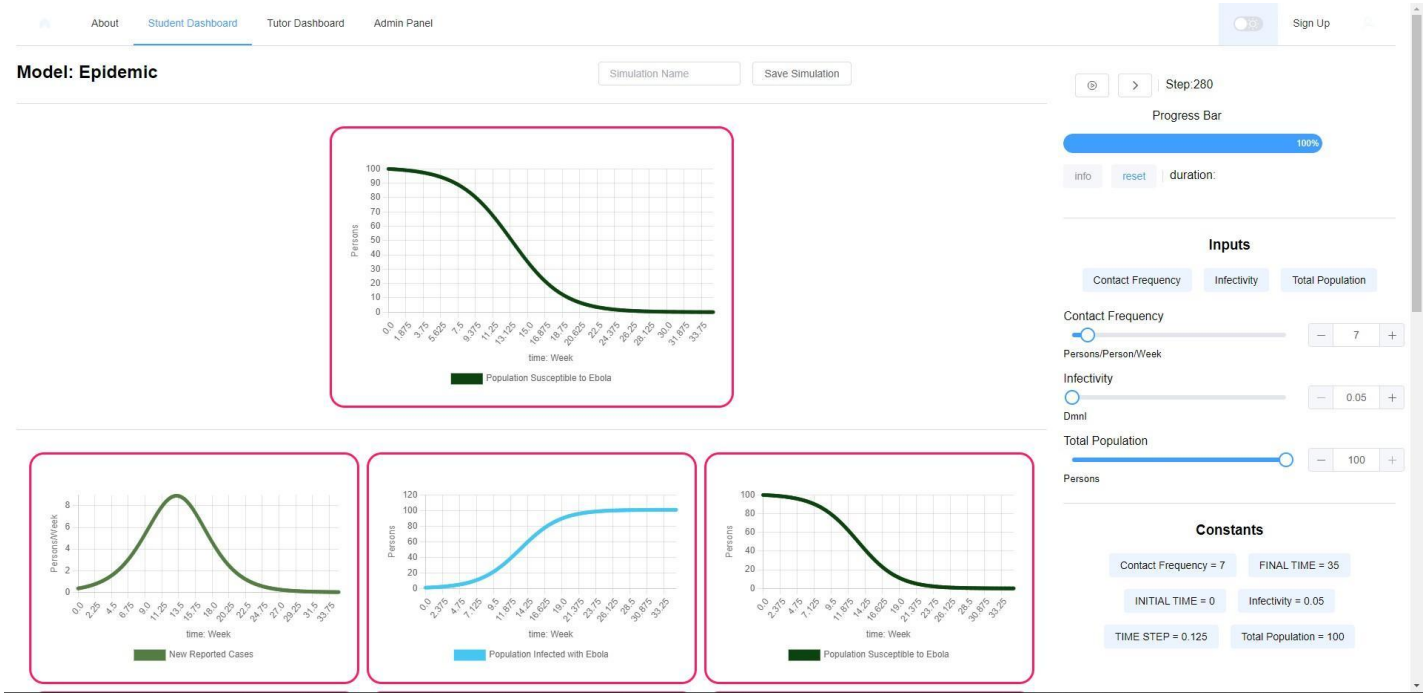

Figure 4.4: Simulation Charts

4.5 Results

Our platform stores all the simulation results in the database in JSON format and by extracting every field, we can get and render information about each variable and stocks. The results are also available for download in a CSV format. In the Following diagrams we present some typical results of a pandemic simulation about ebola disease 4.1.

Table 4.1: Ebola Infection Simulation

Time	New Reported Cases	Population Infected with Ebola	Population Susceptible to Ebola
00:00.0	0.35	1	100
00:00.1	0.365152676	1.04375	99.95625
00:00.2	0.380947081	1.089394084	99.91060592
00:00.4	0.397409117	1.13701247	99.86298753
00:00.5	0.414565619	1.186688609	99.81331139
00:00.6	0.432444373	1.238509312	99.76149069
00:00.8	0.451074144	1.292564858	99.70743514
00:00.9	0.470484693	1.348949126	99.65105087
00:01.0	0.490706803	1.407759713	99.59224029
…	…	…	…
00:05.0	1.806956384	5.400368655	95.59963134
…	…	…	…
00:35.0	0.014052233	100.9602326	0.039767378

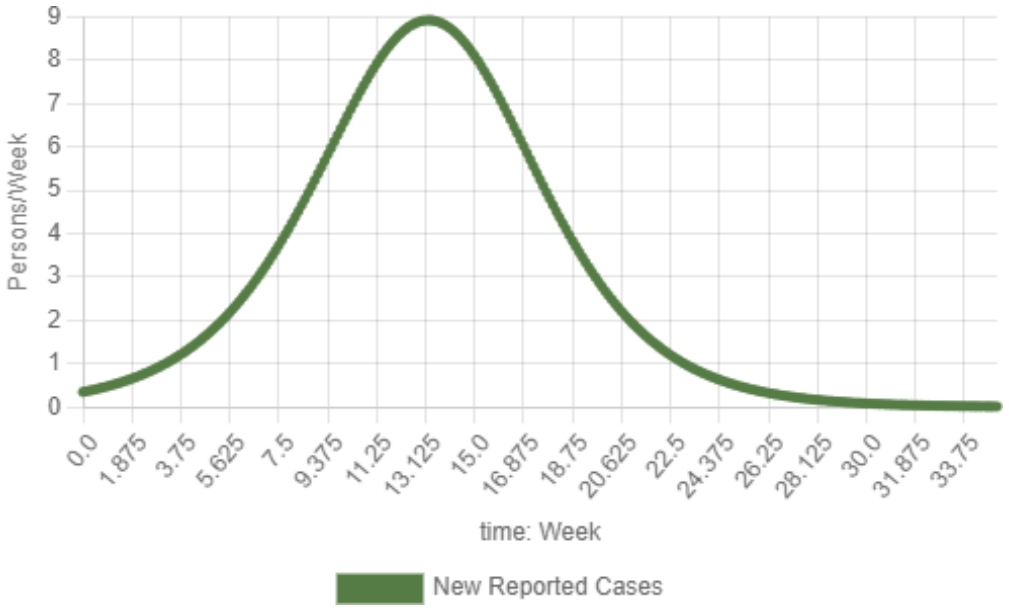

Figure 4.5: New reported cases - Ebola infection

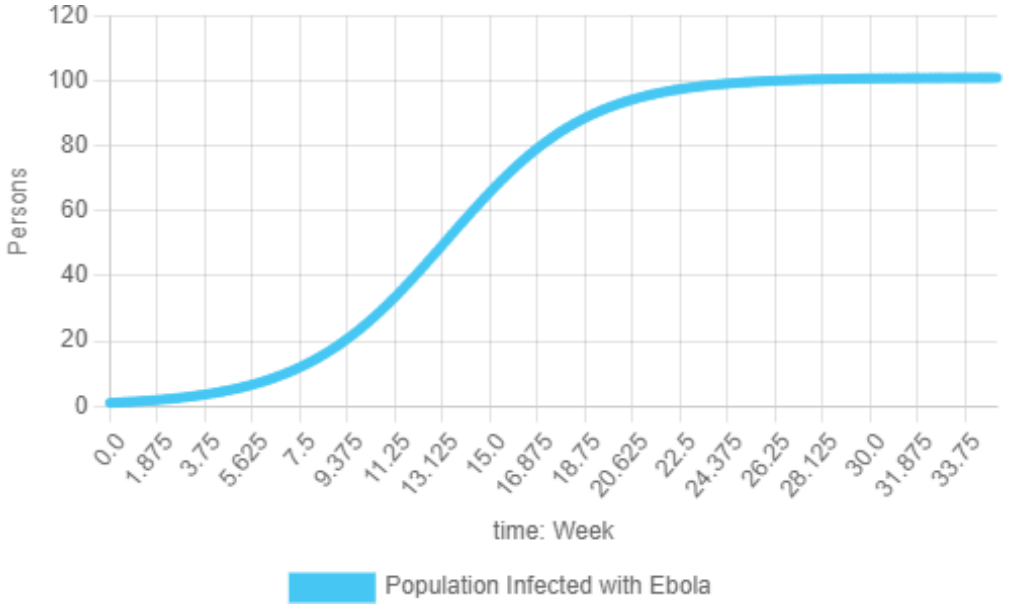

Figure 4.6: Population infected - Ebola infection

Chapter 5

Conclusions

5.1 Conclusions & Summary

In our project, we have created a full-stack prototype web platform for storing, running and managing SD Simulations in a potential e-classroom. We used FastAPI on the back end, PySD as the core engine running the simulations, and Vue 3 as the front end framework, with numerous other small tools for our tasks. The name of our platform is **PySims**, from Python and Simulations. PySims, hopes to offer an intuitive UI giving system dynamists and their students the potential to train beyond their laboratory. This will give future researchers the ability to gather a lot of data in order to reflect on their teaching approach on SD. In the technical scope, the project could become complementary to PySD. Anyone interested in the codebase of this project is kindly asked to communicate with the authors.

5.2 Difficulties

As web development becomes richer, the complexity of creating and maintaining an application from a small number of people or even one, is becoming a difficult task. The modern junior developer should learn a variety of tasks in the whole spectrum, from designing and developing the back end and front end, to developing a database and a UI-UX environment and deploying configuration. This, highlights the need of teamwork and cooperation in the

developers community. Another consideration that was a difficulty, is the variate and diversion of different tools that were used. Sometimes, a tool has its own unique syntax for a specific task, for example on the deployment stage, server configuration on heroku platform was a difficult task. Lastly, for a novice developer it is hard to decide between a dozen different tools and technologies which and when to be used, as often libraries and plug-ins are upgrading and changing very fast.

5.3 Suggestions for Future expansion

As the complexity of this ecosystem goes beyond of this dissertation, we mention some of the future ideas and part of the road map PySims:

- Implement Authentication and connect University's database with auth
- Design User's Profile, with statistical information
- Route guarding and separation of administrators panels
- Examples of users behavioral analysis
- Deploy on University's Server

The profound answer to any data question is what to do with the data? One of the first thoughts of the writer was not only to create a tutor-student relationship with these data, but a holistic class relationship by citing students' approaches between them. To specify, let's imagine an idea from the gaming industry, where a ladder with best results exists. We strongly believe that this would increase the motivation of students in a System Dynamics class. A similar game idea is to assign different roles as sub-models in a bigger one. For example, in the model of local brewery, some students are assigned to decide about first raw materials as the suppliers, others to decide about production as the brewery owners etc. Thus, the complete picture of the ecosystem and real-life simulation problems could be reproduced, letting only the imagination of the modelers define the scenarios.

Another path of development would be the implementation of behavioral analysis of students' strategies, as it was studied by Vleioras[31] where every decision of the user-student is stored and co-related with the results.

This would help both students and tutors to understand their approach, and may lead to a reinforcement learning technique analysis.

Bibliography

[1] Harrison Hao Yang. New world, new learning: Trends and issues of e-learning. *Procedia - Social and Behavioral Sciences*, 77:429–442, 04 2013.

[2] Barr christopher. Businesses play war games. *Pc Mag*, 12(11):31, 06 1993.

[3] When simcity got serious: the story of maxis business simulations and simrefinery. `https://obscuritory.com/sim/when-simcity-got-serious/`. accesed on: 17-09-2022.

[4] Theo Hug. Micro learning and narration. exploring possibilities of utilization of narrations and storytelling for the designing of" micro units" and didactical micro-learning arrangements. In *fourth Media in Transition conference*, volume 6, 2005.

[5] David Gibson, Clark Aldrich, and Marc Prensky. *Games and simulations in online learning: research and development frameworks: research and development frameworks*. IGI Global, 2006.

[6] S Barry Issenberg. The scope of simulation-based healthcare education, 2006.

[7] Vitor Duarte Teodoro. Playing newtonian games with modellus. *Physics Education*, 39(5):421, 2004.

[8] Houghton J and Siegel M. Advanced data analytics for system dynamics models using pysd. In *Proceedings of the 33rd International Conference of the System Dynamics Society*, 2015.

[9] Ann D Cook. *A case study of the manifestations and significance of social presence in a multi-user virtual environment*. PhD thesis, 2009.

[10] Andreas Holzinger, Michael Kickmeier, Siegfried Wassertheurer, and Michael Hessinger. Learning performance with interactive simulations in medical education: Lessons learned from results of learning complex physiological models with the haemodynamics simulator. *Computers & Education*, 52:292–301, 02 2009.

[11] Goodwin J. S. and S. G. Franklin. The beer distribution game: using simulation to teach system. *The Journal of Management Development*, 13(8):7–15, 1994.

[12] Federico Pasin and Hélène Giroux. The impact of a simulation game on operations management education. *Computers & Education*, 57(1):1240–1254, 2011.

[13] Igor S. Mayer. The gaming of policy and the politics of gaming: A review. *Simulation & Gaming*, 40(6):825– 862, 2009.

[14] Jan Ritsema van Eck and Eric Koomen. Characterising urban concentration and land-use diversity in simu- lations of future land use. *The Annals of Regional Science*, 42(1):123–140, Mar 2008.

[15] Meadows Donella H, Meadows Dennis L, Jørgen R, and William W Behrens III. *The Limits to Growth*. New York, Universe Books, New York, 1 edition, 1972.

[16] Wells. Management games and simulation in management development: an introduction. *Journal of Management Development*, 9(2):4–6, 1993.

[17] A J Faria. Business simulations games: current usage levels - an update. *Simulation Gaming*, 29:295–308, 1998.

[18] AJ Faria and John R Dickinson. Simulation gaming for sales management training. *Journal of Management Development*, 1994.

[19] Luigi Proserpio and Dennis A. Gioia. Teaching the virtual generation. *Academy of Management Learning & Education*, 6(1):69–80, 2007.

[20] Stefanos Poultsakis, Stamatios Papadakis, Michail Kalogiannakis, and Sarantos Psycharis. The management of digital learning objects of natural sciences and digital experiment simulation tools by teachers. *Advances in Mobile Learning Educational Research*, 1(2):58–71, Jun. 2021.

[21] Inas Ezz, Cecilia Loureiro-Koechlin, and Lampros Stergioulas. An investigation of the use of simulation tools in management education. In *Proceedings of the 2012 Winter Simulation Conference (WSC)*, pages 1–14. IEEE, 2012.

[22] Jan van der Meij and Ton de Jong. Supporting students' learning with multiple representations in a dynamic simulation-based learning environment. *Learning and instruction*, 16(3):199–212, 2006.

[23] Paul Chandler and John Sweller. Cognitive load theory and the format of instruction. *Cognition and instruc- tion*, 8(4):293–332, 1991.

[24] Hermina JM Tabachneck, Anthony M Leonardo, and Herbert A Simon. How does an expert use a graph? a model of visual and verbal inferencing in economics. In *Proceedings of the Sixteenth Annual conference of the Cognitive Science Society*, pages 842–847. Routledge, 2019.

[25] Robert Kozma. The material features of multiple representations and their cognitive and social affordances for science understanding. *Learning and instruction*, 13(2):205–226, 2003.

[26] Society for learning analytics research. `http://www.solaresearch.org/about/`, April 2012.

[27] Eberlein, Robert, Chichakly, and Karim. Xmile: A new standard for system dynamics. *System Dynamics Review*, 29, 07 2013.

[28] Houghton, James, Siegel, and Michael. Advanced data analytics for system dynamics models using pysd. In *Proceedings of the 33rd International Conference of the System Dynamics Society*, 2015.

[29] Stack overflow yearly survey 2021. `https://insights.stackoverflow.com/survey/2021`. accesed on: 21-08-2022.

[30] Vite documentation definition. `https://vitejs.dev/guide/#overview`. accesed on: 21-08-2022.

[31] Alkiviadis Vleioras. Modeling of decision making and learning in a business simulation game. Master Dissertation, May 2020. New Entrepreneurship, Innovation and Development Department of Planning and Regional Development Department of Mechanical Engineering Department of Economics University of Thessaly.

Chapter

Appendix

1 Example of run_simul function

```
1
2
3  def run_simul(db:Session , model_details : schema.Simul_post, step_run : bool) :
4
5      fileDir = f ' ./ models/{ model_details . model_name}'
6
7      if ( not path . exists ( fileDir ) ) :
8          raise Exception("There is not such a name model")
9
10     fileExt = r '*.py'
11     model_path = list ( pathlib . Path( fileDir ) . glob( fileExt
12     ) ) if (model_path):
13         model = pysd. load(model_path[0])
14     else :
15         fileExt = r '*.mdl'
```

```
16      model_path = list ( pathlib . Path( fileDir ) . glob( fileExt ) )
17      model = pysd.read_vensim(model_path)
18
19   #cur_step as an integer −        (0...1..2... N)
20  if ( step_run ) :
21      cur_step = int (( model_details . end_time − model[”INITIAL TIME”] − model[”TIME STEP”])/model[”
            TIME STEP”])
22  else :
23      cur_step = 0
24
25  print ( f ’ cur_step is :{ cur_step}’)
26
27  # Here starts  the model run part
28  if ( step_run ) :
29      if ( cur_step > 0) :
30          print ( f ’N_th run of Simulation :({ model_details . start_time }, { model_details . end_time})’)
31          df = model.run( initial_condition =” ./ user / results / pickles / final_state . pic”, return_timestamps =(
                model_details . end_time), params=(model_details . params))
32
33      else :
34          print ( f ’1 st Run of Simulation :({ model[”INITIAL TIME”]},{model[”TIME STEP”]})’)
35          df = model.run(params=(model_details . params), return_timestamps =(model[”INITIAL
                TIME”],model[” TIME STEP”]))
36
37  else :
38      print ( f ’ model_details params are : { model_details . params}’)
39      if ( cur_step>0): #cur_step > 0
40          df = model.run( initial_condition =” ./ user / results / pickles / final_state . pic”, params=(model_details
                . params))
```

```
41          else :
42              df = model.run(params=(model_details . params))
43
44      # Output Part
45      os. makedirs(f' ./ user / results / pickles / ' , exist_ok=True)
46      model.export(" ./ user / results / pickles / final_state . pic")
47
48      if ( cur_step>0):
49          data_as_dict = df . to_dict () # storing for merging in step run
50          f = open(f' ./ user / results / simulation_state . json ' )
51          dict_before = json . load( f )
52          result = mergeStepDicts( dict_before , data_as_dict )
53      else :
54          data_as_dict = df . to_dict () # storing for merging in step run
55          result = data_as_dict
56
57
58      with open(f' ./ user / results / simulation_state . json ' , 'w') as convert_file :
59          convert_file . write ( json . dumps( result ) )
60
61          # create db entry
62
63
64      print ( f ' cur_step={cur_step}')
65
66      simulation_res = models.Simulation (simulation_name= model_details . simulation_name,
67      model_name = model_details.model_name,
68      csv_path = None,
69      json_data = json . dumps( result ) ,
```

```
70      params = model_details . params
71      )
72
73      return ( simulation_res )
```

2 API Directory with description

- `.env` – the folder containing all the necessary packages and environmental variables
- `.gitignore` – we add all the files we don't want to upload in github
- `alembic.ini` – stores all the database record, and migration
- `app.py` – our main driver, here all the paths endpoints are defined
- `crud.py` – all the functions that are called from app.py
- `database.py` – the SQLAlchemy engine and database connection
- `db-init.py` – an initialization scripts, which runs only for once to upload the models on the database
- `foo.db` – this is a test database for development purposes
- `functions.py` – helpful functions
- `models.py` – the models classes defines the tables in the database
- `Procfile` – this file is needed for deployment, run's the server
- `readme.md` – just a read me file in markdown
- `requirements.txt` – includes all the necessary packages name, it is the equivalent of packages.json in js

- `schema.py` – contains the various schemas of our api
- `__init__.py` – an empty file with special functionality in python to declare the root file as a package.

You can find the complete PySims API documentation in https://pysims-github.herokuapp.com/docs