Group #1 (as per the assignment)

An IDE for Atlas Thing Architecture

Harouche, Lior

Lisiansky, Doron

Lora, Edwin

Shmul, Daniel

Welsh, Logan

Replace the photo above with a main photo of your IDE (of your choice)

1. Introduction **(about 1 pages)**

In our project, we built an IDE for managing things in an IoT environment and invoking the things' services (from this point we will be reffering to it as IDE). The IDE we developed is based on the Atlas IoT middleware framework. The IDE was written in Python, and we used PYQT5 (an open source Python GUI framework) to program and visualize the GUI of the IDE.
The purpose of the IDE is to help users develop and test IoT applications and examine the functionality of the users' Iot Things.
The IDE is setting up a connection with the VSS and is able to read tweets and recognize things in the VSS. Through reading the tweets, the IDE is able to present the user the things in the VSS, their corresponding services, the relationships between the different services and even offer the user to create an app that uses said services (using the recipe tab). As requested in the project description, our IDE has 5 tabs: Things, Services, Relationships, Recipe, Apps:

- **Things** - lists all the things that are connected to the VSS and displays basic information about them.
- **Services** - lists all the available services, and the corresponding thing that can offer said service. The tab also offers a filter to show services from specific things.

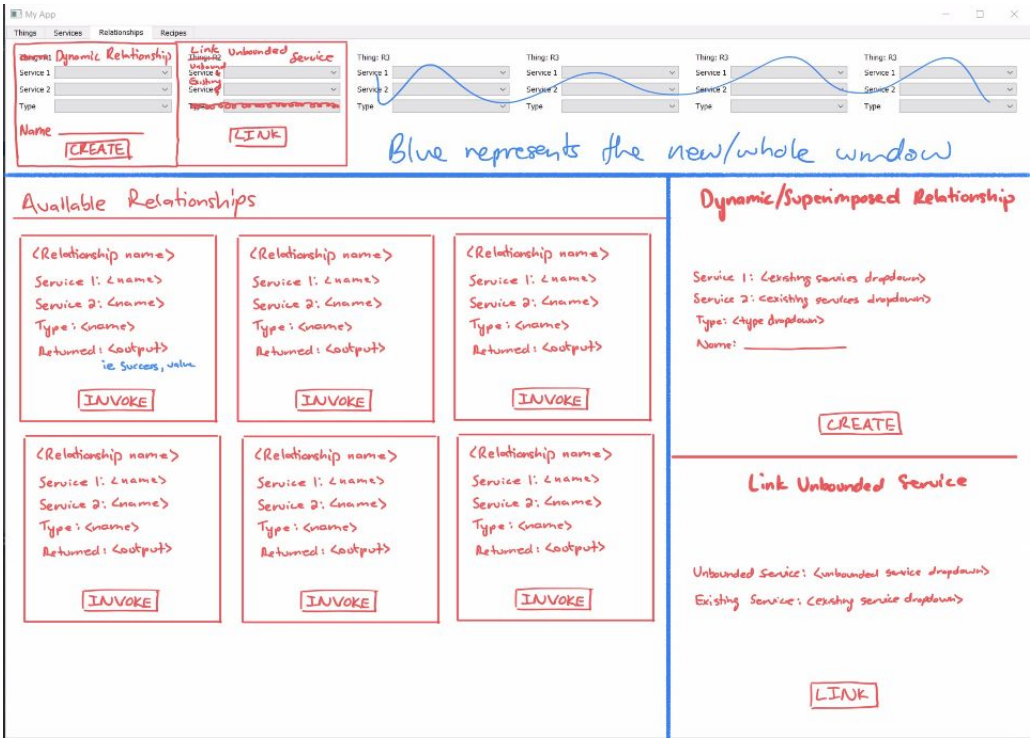
- **Relationships** - this tab displays all the relationships between the things and the services as defined in the XML-DDL file. It also allows the user to create new relationships and bind unbound relationships.
- **Recipe** - this tab allows the user to create a new IoT application based on the things and their available services. The app is created through the GUI using clicking operations.
- **Apps** - this tab displays existing applications and helps the user to manage them

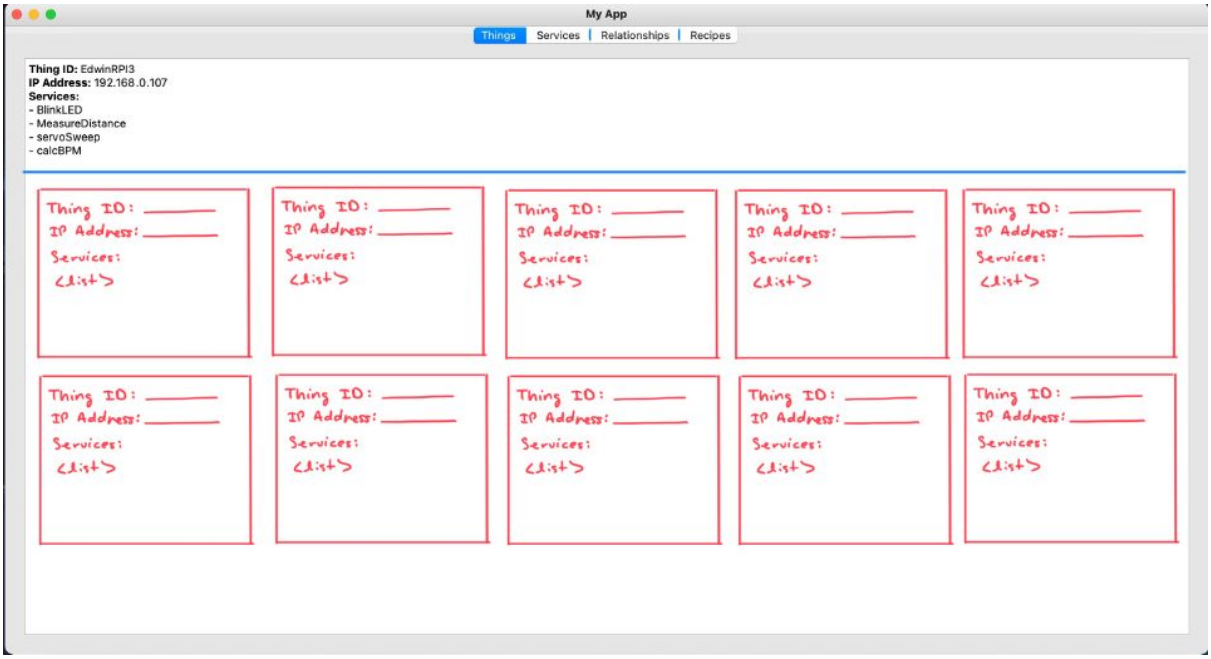
In order to demonstrate our IDE and provide better understanding of it, we've created a demo that explains it. The demo can be viewed in the following YouTube link:

Provide a link to your YouTube Video here in the introduction.

2. Project Design **(about 1.5 pages)**

Here you will **first describe the design of your project** mainly a chart that describes the overall system architecture in terms of components and their inter-relationships. You should provide an architecture chart plus textual description of its components. You should discuss how the system was intended to be used. Mention any assumptions made.





2.1 Challenges Faced **(about 1 page)**

You should discuss the limitations and difficulties faced clearly and factually without exaggeration or dramatization. You may lose points if you dramatize without adequate explanations or justifications. This section should be useful to a follow-on group that may attempt to build on what you have accomplished and take the project to a next iteration of improvements. So, write this section with this “purpose” in mind.

3. Implementation

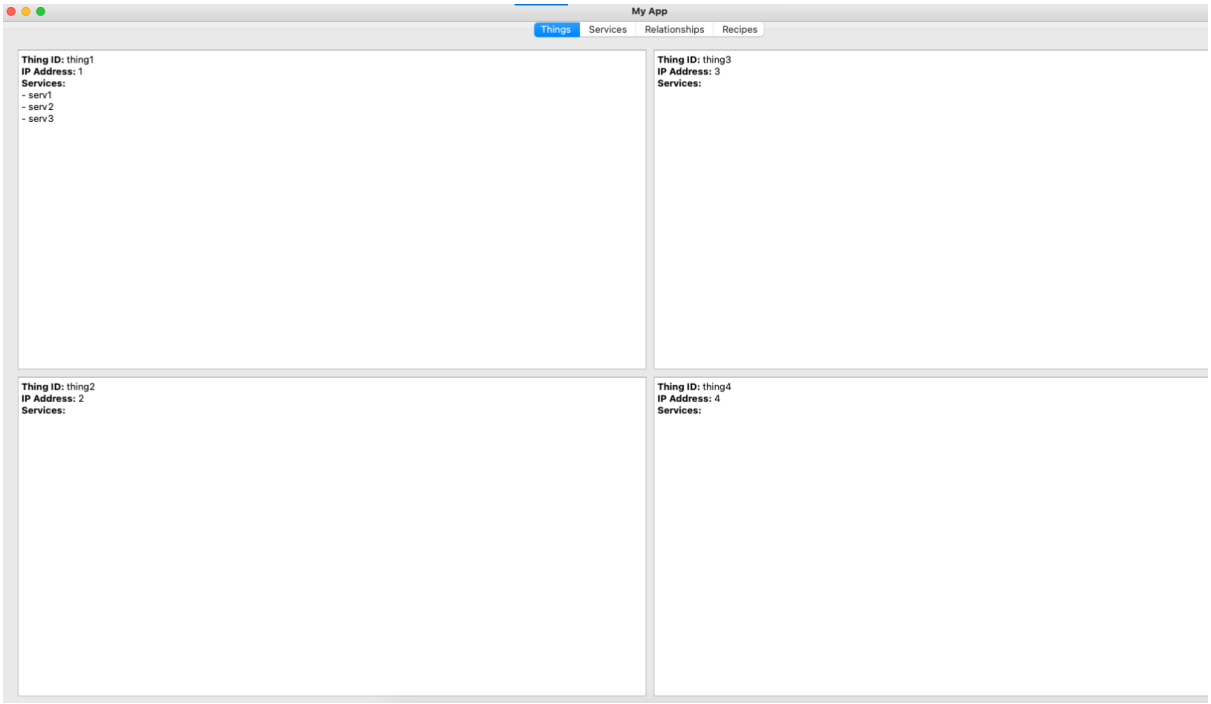
GitHub link - https://github.com/dshmul/IoTProject.git

we will give information of the designed this Atlas IoT application IDE, including all the features we have implemented, the tools we used in the backend/frontend.

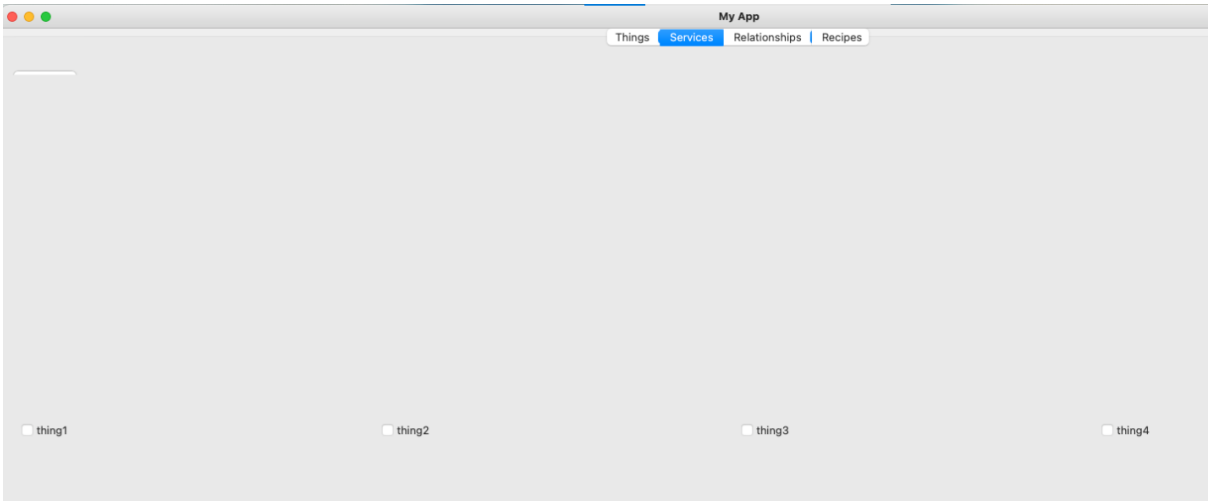
we used mainly python for the implementation of all the features, for the ide itself we used pyqt5 library for connecting the frontend and the backend.

init - in the init file we are listening to tweets and the ide filters the received tweets, we are parsing and extracting the tweets according to different types of tweets And then saving the essential information like services,things and relationships into relevant data structures for future need and for the different tabs and applications.

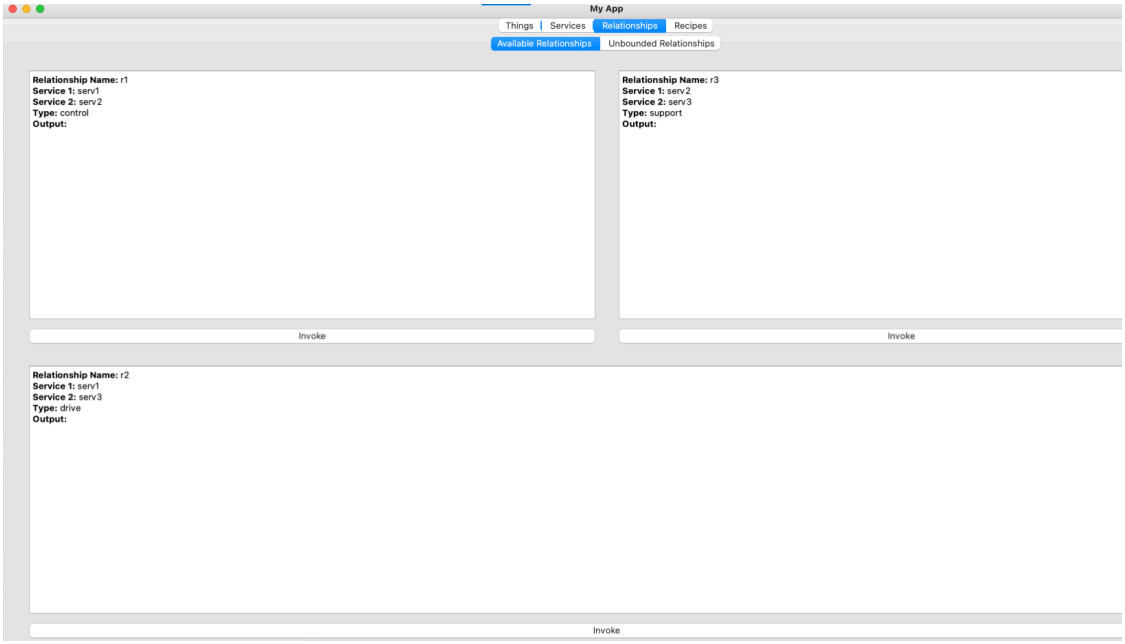
Things tab - shows the thing names, the IP addresses of the pi's.for displaying the Things tab we are using the data that stored inside the things data structure which consist: thing name, thing IP and list of services.



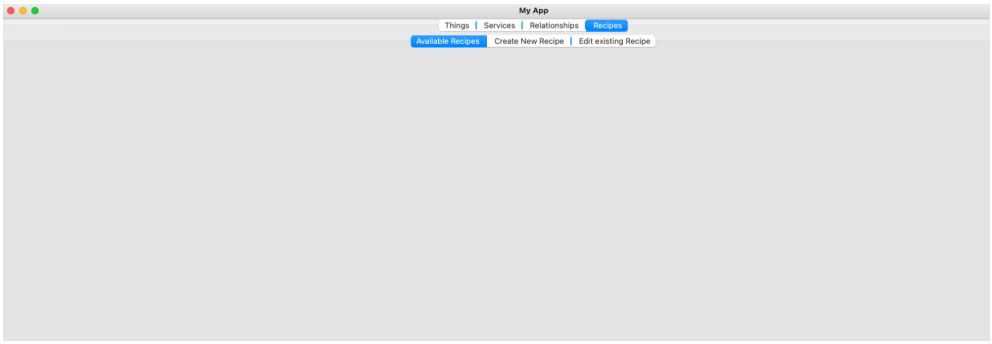
Services tab: the service tab lists all the existing services available in our VSS and its corresponding thing. first we have a list of all the things in our VSS, after choosing a thing there is a list of its corresponding services using Service data structure that consist of the service name, a list of relationships, the corresponding thing, and if the service is active or not for future needs, there is a filter to find a specific service.



Relationships tab - Shows all relationships using a filter similar to that of the service. there is two types of relationships under this tab, there is list of all available relationships that we can use, and there is a list of all unbounded relationships which containing at least one unbound service, we use the Relationship data structure that consist of the Relationship name , the two services that share the relationship, the type of the relationship and a boolean for each service to tell the developer if the service is unbounded or not, we use unbounded service data structure to save the services that are currently unbounded and later link them an existing service and making the relationship available .



Recipe tab - this tab Handles the recipes to create apps. under this tab we have 3 more tabs, the available recipes tab which displaying all the available apps we can use and run, create new recipe tab which open a framework for creating new apps/recipes, edit existing recipe tab which the user can change/delete and save existing apps.



Application Manager - implements the following functionalities : Save, Upload, Activate, Stop, and Delete.

- save - save button is presented under the create new recipe and edit existing recipe tabs, after editing or creating a new recipe, save button will present the new app in the available recipes for activation or future use.

- upload - upload button is presented under the available recipes tabs, after clicking the button there is option to decide from which file we will upload the app and present the uploaded app in the available recipes for activation or future use.
- activate - activate button is presented under the available recipes tabs, after clicking the button the chosen app will start to run.
- stop - stop button is presented under the available recipes tabs, after clicking the button from an active app it will stop immediately .
- delete - delete button is presented under the available recipes tabs, after clicking the button from an app it will immediately delete the app from all the memory locations and the app will disappear from the apps list .

3.1 Subjective Evaluation (0.5 page)

Overall, this project was very challenging yet very satisfying since we managed to create an Atlas IoT IDE. We were very pleased to have managed to implement all the tabs and allow the user to use this tool to experiment and create an Atlas application. It was also a great experience for all of us to work with python (and PyQT specifically) and experience a tough group project in which the work needed to be divided. It was a challenge to come up with the design to implement all the required specifications, and working with a formerly unknown python GUI tool such as PyQT. However, despite these challenges, we managed to come up with a final product.

It would have been very useful for this project if the expectations of the layout of the IDE were more specific, for example by giving screenshots of an existing IDE of how the tabs are supposed to look like. In addition, a Q&A sheet from previous years of this project could have saved a lot of unnecessary trials and errors. Also, an appendix with reading material regarding necessary information relevant to creating the IDE (such as Backus-Naur for the recipe tab) would have been helpful. But, overall, the instructions were clear and we were able to finish the project.

3.2 Future Work (0.5 pages)

Although our IDE is very functional and offers a significant amount of tools for the user to develop IoT applications, some further work needs to be done to better this project:

- Using PyQT was challenging. Perhaps for future groups it would be wiser to choose a better GUI tool such as Visual Studio
- A better and more detailed Recipe tab can be built to offer better and more complex apps.
- We used merely a basic design tool. In order to take this project to another level and provide the user with the best experience, a more advanced design tool can be used.
- We used only one thread to listen to the tweets. This might create an issue if several things are online and there is a collision between different tweets from

different things. In order to solve this issue, a better approach would be to create a single thread to listen to tweets from each thing.

Although further work still needs to be done, this IDE is still very functional and allows the user to explore the services in an IoT environment and create his own Atlas applications using the IDE.

3.3 Distribution of Effort **(who did what, names listed alphabetically)**

	Lior Harouche	Doron Lisiansky	Edwin Lora	Daniel Shmul	Logan Welsh	Total
Service Code	20%	20%	20%	20%	20%	100%
Design	25%	25%	25%	25%		100%
Frontend			25%		75%	100%
Backend						100%
Testing	30%	20%	20%	15%	15%	100%
Report	30%	30%	13%	13%	13%	100%