# My own Projects and Work

## Student Researcher - Chemical Technologies @ RDM Kade

**Author:** Ruben Flinterman

One of my jobs is as a student researcher for Chemical Technologies at RDM kade (also Hogeschool Rotterdam).
This job began as "Project 5/6" which was part of Technical Computer Science.

**A brief summary of what Chemical Technologies does:**
They develop and improve a mix of chemicals based on certain fruit sugars which changes color if there is a certain chemical nearby.

**A brief summary of what Niko van Ommen and I do:**
Niko and I develop the software which links the color change to an intensity. We do this by creating our own hardware composition with custom written software and a mobile app.

**The goal:**
The goal is to create and deliver a good research paper for IEEE of which everyone involved is a co-author, including Niko and me.
**What do I do?:** I currently work on the second version of our flutter application, so I develop in Java and Dart using gradle as the build tool of choice.
The application is repeatedly tested on Android, although they are not yet automated. This is on the roadmap, after I finish the redo the Bluetooth implementation.

The second version means that it is not written based on the previous version but rather an actual version instead of a rough prototype.
In my version I focused on better iOS implementation, a notification system and best of all a better UI with better performance.
This is why I keep replacing and removing unnecessary code so the amount of lines written decreases and the simplicity and documentation quality stays high.

An example of code, in this case for our navigation, is:

```
class _MyHomePageState extends State<MyHomePage> {
  // Start app on home page (1) and keep tabs on the current page
  int currentPageIndex = 1;

  // Select the page content with switch cases
  Widget _getPage(int page) {
    switch (page) {
      case 0:
        return const ConnectPage();
      case 1:
```

```
        return const HomePage();
      case 2:
        return const InfoPage();
      default:
        return const HomePage();
    }
  }

  @override
  // Build the navigation bar with the logic to switch pages if the
correct button is pressed
  Widget build(BuildContext context) {
    return Scaffold(
      bottomNavigationBar: NavigationBar(
        // On navigation select
        onDestinationSelected: (int index) {
          // Set correct page by index
          setState(() {
            currentPageIndex = index;
          });
        },
        // Give the navigation a specific style
        backgroundColor: HexColor("#EBEBEB"),
        indicatorColor: HexColor("#D8DBE2"),
        // Highlight the button from the current page
        selectedIndex: currentPageIndex,
        // Add the buttons which we need to press to change pages
        destinations: const <Widget>[
          NavigationDestination(
            selectedIcon: Icon(Icons.add_circle),
            icon: Icon(Icons.add_circle_outline),
            label: 'Connect',
          ),
          NavigationDestination(
            selectedIcon: Icon(Icons.home),
            icon: Icon(Icons.home_outlined),
            label: 'Home',
          ),
          NavigationDestination(
            selectedIcon: Icon(Icons.info),
            icon: Icon(Icons.info_outline),
            label: 'Info',
          ),
        ],
      ),
      // Set the new page content
      body: _getPage(currentPageIndex),
    );
  }
}
```

These lines of code give us the ability to load page content from other files from within the navigation.

The selected page will also be highlighted in the navigation and is easy to change.

## Learning Kubernetes

**Author:** Ruben Flinterman

As an hobby and as of 20-11-23 also for work (datalab) I research Kubernetes and try to get it working for a production environment where it needs to be able to scale easily, on a bare-metal server.

Although kubernetes is not a programming language it does provide a few things regarding experience which could be important for a developer (with an eye for writing safe software):
1) How you can both easily and safely configure services, so it won't give any issues on long term (IP, ports, subnets)
2) It gives you a general understanding of networking
3) It gives you a general understanding of yaml and bash scripts
4) It forces you to properly think-out your infrastructure. Ex: Do you want a database next to your web application? Do they need to run on the same cluster, and if so, do they need to be in the same pod?

## Class based rewrite of Challenge Week 1 in my spare time

In some of my spare time a friend (from Technical Computer Science) and
I challenged each other to quickly come up with a game concept while still following the guidelines of challenge 1.
As we are both pretty experienced with writing software (we did projects with each-other before) we did add some rules of our own.

**The additional rules:**

```
- AI is not allowed unless you use it for JSON.
- It can't have errors.
- You can't copy any existing work (Websites like stackoverflow are
allowed)
- You can use libraries as long as it's build into Python.
```

**The normal rules:**

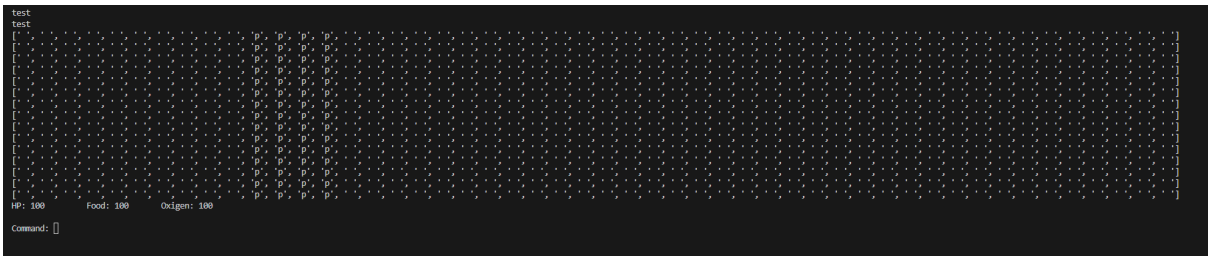
```
Minimal requirements:
- 10 locations
- 10 items
- 5 puzzles
- All python techniques of week 1-3
- No errors
- No copies of work from fellow students or adventures found on the
internet
- Theme must be SPACE TRAVEL

Nice to have:
- Images (look at the turtle library in python)
- Sounds (depends on your operating system)
- Understand commands like GO WEST, OPEN THE DOOR, PICK UP LASER …..
```

Obviously, for a challenge for a small evening we didn't do 10 locations, 10 items or 5 puzzles. This much wouldn't be reasonable in a few hours.

**After the challenge:**
This resulted into a fun challenge from around 2 hours after which we stopped (we both didn't finish).
Why we thought it was fun is because even though we had a lot of rules we both came up with a whole other concept.
He went for a 'Super Mario Bros. (1985)' styled game but then based completely on ASCII.
I went with the game concept we used during the challenge week, so I could look how much time I would need to write a game
with the same core concepts (so ignoring stuff like a story). So a 'Space Quest I' like command system but without any real visuals.

Some of the screenshots we shared early in the evening were:

His Super Mario Bros like game:




My game, a Space Quest like game:

```
================================================================================
You start in the game room.
Description: A spacious game room with various games and entertainment.
================================================================================
What do you want to do?
```

You can find my code on GitHub:
https://github.com/RFlintstone/text-based-adventure-python/blob/main/main.py

I made `World` a parent class, `Player` is a child of `World` as it is 'standing' in the world, and then we also have a class called `Console`.

`World`: Responsible for changing rooms
`Player`: Responsible for the player actions and inventory
`Console`: Responsible for the screen width so we have some sort of implementation for a resolution.

```python
# Our Classes
c = Console()
w = World()

# Prettify the console
print("=" * c.get_screen())

# Ask for input
p = Player(input("What's your name? "))

# Add dummy item so we can test the inventory
p.add_inv("key")

# Set our first room
w.set_room("game room")

# Prettify the console
print("=" * c.get_screen())

# Give our first starting location
print(f"You start in the {w.get_room()}.")
print(f"Description: {w.load_room()}")

while True:
    # Every time we will be asked a command we also prettify the
console
    print("=" * c.get_screen())
    cmd, arg = p.command(input("What do you want to do? "))

    # Choose an action based on the tuple from 'p.command'
    if cmd == "go":
        # Go to a different room, if it exists
```

```
            p.move(arg)
        elif cmd == "use":
            # Use an item, the item would be removed from the inventory
            p.use(arg)
        elif cmd == "inspect":
            # Inspect things from a room
            res = p.inspect(arg)
            # Then print the result of our inspection
            print(res)
        elif cmd == "talk":
            # Talk to someone - I ignored this in the 1v1 challenge because
    of time constraints
            p.talk(arg)
```