

Student: Justin Deleon
Mentor: Gummi Traustason and Joseph Quinn
Instructor/Faculty: **Dr. Masoud Sadjadi**, Florida International University

```
See 'snap info docker' for additional versions.

# python3 docker-VirtualBox:~/Desktop/Skillcourse/skillcourt-websites docker compose exec webapp pmprx
prisma up --push
[ERROR] [PRC]: permission denied, open /app/.top.13.94296606eb6dbda441780625c878006e:
# python3 docker-VirtualBox:~/Desktop/Skillcourse/skillcourt-websites docker compose exec webapp pmprx
prisma up --push
# python3 docker-VirtualBox:~/Desktop/Skillcourse/skillcourt-websites docker compose exec webapp pmprx
prisma up --seed

Command 'docker-compose' not found, but can be installed with:

sudo snap install docker # version 19.19.17, or
sudo apt install docker-compose # version 1.25.0-1
```

```
graph TD; View["View  
Flutter Application's game statistics"] -- "shows user statistics" --> Model["Model"]; Model -- "queries or updates information" --> Database[(Database)]; Database -- "gets or updates pad information" --> Controller["Controller  
Flutter Application"]; Controller -- "Pad to App" --> Users["Users  
coach or players"]; Users -- "displays game statistics" --> View;
```

The diagram illustrates the MVC (Model-View-Controller) pattern for a game application. It consists of the following components and interactions:

- View** (Flutter Application's game statistics): Displays game statistics to the users.
- Users** (coach or players): Interacts with the application via a "Pad to App".
- Controller** (Flutter Application): Receives input from the users and sends queries or updates to the database.
- Model**: Manages the application's data and logic, receiving queries or updates from the database and sending user statistics to the view.
- Database**: Stores application data and provides information to the model.

The image displays three mobile application screens for a station creation form, all featuring a dark green background and white text.

- Screen 1 (Left): Sign In**
 - Header: "Sign In"
 - Fields: "Email" and "Password"
 - Buttons: "LOGIN" and "SIGN UP"
- Screen 2 (Middle): Create user**
 - Header: "Create user"
 - Fields: "Enter Username", "Enter First Name", "Enter Last Name", "Email", "Date of Birth", "Phone Number" (with a red asterisk indicating a required field), "Password", and "Confirm Password"
 - Button: "SIGN UP"
- Screen 3 (Right): Station Creation Form**
 - Header: "Station Creation Form"
 - Fields: "Station Name", "Select Object" (with a dropdown arrow), and "Description of Station"
 - Buttons: "Undo", "Save", and "Reset"

Each screen includes a status bar at the top showing the time (8:51) and various system icons. The bottom of each screen displays the copyright notice: "© 2021 AllRights Reserved".

```
graph LR
    User((User))
    Pad((Pad))
    StartGame([Start Game])
    ConnectPad([Connect Pad])
    ViewPad([View pad])
    RegisterPad([Register Pad])
    Database((Database))

    User --> StartGame
    User --> ViewPad
    Pad --> RegisterPad
    RegisterPad --> ViewPad
    ViewPad --> ConnectPad
    ConnectPad --> Database
    Database --> StartGame
```

The diagram illustrates the interactions between a User, a Pad, and the Pad Game system. The User can initiate the game (Start Game) or view the pad (View pad). The Pad can register (Register Pad), which then leads to viewing the pad (View pad). The View pad use case connects to the Connect Pad use case, which in turn interacts with the Database. The Database also provides information back to the Start Game use case.

Overall, this capstone 2 project was a great learning experience that I will take with me into the professional field. The product owner, Gummi, was a pleasure to work with, and is now in the process of bringing this app forward to be used by a big Florida soccer institution.

- Java JRE > 1.8
- A GitHub Account
- Git
- Git-flow-avh
- Flutter SDK
- A phone emulator like Android Studio
- Android Tool Chain
- Python3
- VSCode (optional)
- A GitHub Account
- Node >= v19.0.0
- Npm >= 9.3.1
- PNPM >= v6.21.0 or a V.M. with Linux installed
- Docker Desktop with PostgreSQL
- Vim (optional)
- VSCode (optional)

The screenshot displays the Docker Desktop application. At the top, there's a navigation bar with 'Docker Desktop' and 'Update your app' buttons. Below this is a 'Containers' tab, which shows a list of running containers. The list includes columns for Name, Image, Status, Ports, Started, and Actions. The containers listed are 'integrated_analyzer', 'postgres', 'allauth-website', 'database1', and 'webapp1'. Below the containers list, there's a terminal window showing the output of a 'docker-compose up' command. The terminal output shows the start of several services: 'integrated-analyzer', 'postgres', 'allauth-website', 'database1', and 'webapp1'. The terminal also shows the output of a 'docker-compose logs' command, which displays the logs for the 'integrated-analyzer' service. The logs show the service starting up and then crashing with a 'TypeError: Cannot read property 'get' of undefined' error. The terminal window is titled 'allauth-website - Visual Studio Code'.

Name	Image	Status	Ports	Started	Actions
integrated_analyzer integrated-analyzer	docker/integrated-analyzer	Running	80.80 TC	13 days ago	Stop Restart Kill Logs
postgres postgres	postgres	Running	5432:5432 TC	31 minutes ago	Stop Restart Kill Logs
allauth-website	postgres:alpine	Running (1/2)			Stop Restart Kill Logs
database1 allauth-database1	postgres:alpine	Running		5 minutes ago	Stop Restart Kill Logs
webapp1 allauth-website-webapp	allauth-website-webapp	Restarting (1)	4300:4300 TC		Stop Restart Kill Logs

Showing 5 items

Showing 5 items

Showing 5 items