1.

package main

import "fmt"

func main(){

sayHello ("Mahadi Chodna")

}

2.

package main

import "fmt"

func main(){

sayHello("Mahadi Chodna")

}

The reason you need a go.mod file even though both code files are in the same folder with the same package name is because the Go compiler needs to know which file contains the main entry point for your program. When you have multiple files with the same package name, the compiler doesn't have a clear indication of which file should be executed first.

The go.mod file helps resolve this ambiguity by providing a project-level configuration that informs the compiler about the module structure and the main entry point. It acts as a central reference for managing dependencies and identifying the primary file for execution.

In the absence of a go.mod file, the compiler may encounter errors or uncertainties when attempting to determine the main entry point, especially when multiple files share the same package name. This is why the error message "gopls was not able to find modules in your workspace" appears.

Therefore, introducing a go.mod file serves two crucial purposes:

1. \*\*Dependency Management:\*\* It enables the Go compiler to identify and manage the external packages or libraries your project relies on.

2. \*\*Main Entry Point Identification:\*\* It clearly defines which file contains the main entry point for your program, resolving any ambiguity when multiple files share the same package name.

In summary, the go.mod file is essential for ensuring smooth compilation and execution of your Go project, especially when dealing with multiple files and dependencies. It provides a structured approach to dependency management and main entry point identification, making the development process more streamlined and efficient.