```
--- AccountBoundary.py ---
from discord.ext import commands
from control.AccountControl import AccountControl


class AccountBoundary(commands.Cog):
    def __init__(self):
        self.control = AccountControl()  # Initialize control object

    @commands.command(name="fetch_all_accounts")
    async def fetch_all_accounts(self, ctx):
        await ctx.send("Command recognized, passing data to control.")

        # Pass the command to the control object
        commandToPass = "fetch_all_accounts"
        result = self.control.receive_command(commandToPass)

        # Send the result (prepared by control) back to the user
        await ctx.send(result)



    @commands.command(name="fetch_account_by_website")
    async def fetch_account_by_website(self, ctx, website: str):
        await ctx.send(f"Command recognized, passing data to control for website {website}.")

        # Pass the command and website to control
        commandToPass = "fetch_account_by_website"
        result = self.control.receive_command(commandToPass, website)
```

```
        # Send the result (prepared by control) back to the user
        await ctx.send(result)


    @commands.command(name="add_account")
    async def add_account(self, ctx, username: str, password: str, website: str):
        await ctx.send("Command recognized, passing data to control.")

        # Pass the command and account details to control
        commandToPass = "add_account"
        result = self.control.receive_command(commandToPass, username, password, website)

        # Send the result (prepared by control) back to the user
        await ctx.send(result)


    @commands.command(name="delete_account")
    async def delete_account(self, ctx, account_id: int):
            await ctx.send(f"Command recognized, passing data to control to delete account with ID {account_id}.")

        # Pass the command and account ID to control
        commandToPass = "delete_account"
        result = self.control.receive_command(commandToPass, account_id)

        # Send the result (prepared by control) back to the user
```

```
        await ctx.send(result)



--- AvailabilityBoundary.py ---
from discord.ext import commands
from control.AvailabilityControl import AvailabilityControl

class AvailabilityBoundary(commands.Cog):
    def __init__(self):
        # Initialize control objects directly
        self.availability_control = AvailabilityControl()

    @commands.command(name="check_availability")
    async def check_availability(self, ctx, url: str, date_str=None):
        await ctx.send("Command recognized, passing data to control.")

        # Pass the command and data to the control layer using receive_command
        command_to_pass = "check_availability"
        result = await self.availability_control.receive_command(command_to_pass, url, date_str)

        # Send the result back to the user
        await ctx.send(result)

    @commands.command(name="monitor_availability")
    async def monitor_availability(self, ctx, url: str, date_str=None, frequency: int = 15):
        await ctx.send("Command recognized, passing data to control.")
```

```
        # Pass the command and data to the control layer using receive_command
        command_to_pass = "monitor_availability"
          response = await self.availability_control.receive_command(command_to_pass, url, date_str,
frequency)

        # Send the result back to the user
        await ctx.send(response)

    @commands.command(name="stop_monitoring_availability")
    async def stop_monitoring(self, ctx):
        await ctx.send("Command recognized, passing data to control.")

        # Pass the command to the control layer using receive_command
        command_to_pass = "stop_monitoring_availability"
        response = self.availability_control.receive_command(command_to_pass)

        # Send the result back to the user
        await ctx.send(response)



--- BrowserBoundary.py ---
from discord.ext import commands
from control.BrowserControl import BrowserControl

class BrowserBoundary(commands.Cog):
    def __init__(self):
        self.browser_control = BrowserControl() # Initialize the control object
```

```
    @commands.command(name='launch_browser')
    async def launch_browser(self, ctx):
        # Inform the user that the command is recognized
        await ctx.send("Command recognized, passing the data to control object.")

        commandToPass = "launch_browser"
        result = self.browser_control.receive_command(commandToPass)    # Pass data to the control
object
        await ctx.send(result)  # Send the result back to the user

    @commands.command(name="close_browser")
    async def stop_bot(self, ctx):
        # Inform the user that the command is recognized
        await ctx.send("Command recognized, passing the data to control object.")

        commandToPass = "close_browser"
        result = self.browser_control.receive_command(commandToPass)    # Pass data to the control
object
        await ctx.send(result)  # Send the result back to the user


--- HelpBoundary.py ---
from discord.ext import commands
from control.HelpControl import HelpControl

class HelpBoundary(commands.Cog):
```

```
    def __init__(self):
        self.control = HelpControl()  # Initialize control object

    @commands.command(name="project_help")
    async def project_help(self, ctx):
        await ctx.send("Command recognized, passing data to control.")

        # Pass the command to the control object
        commandToPass = "project_help"
        response = self.control.receive_command(commandToPass)

        # Send the response back to the user
        await ctx.send(response)
```

--- LoginBoundary.py ---

```
from discord.ext import commands
from control.LoginControl import LoginControl

class LoginBoundary(commands.Cog):
    def __init__(self):
        self.login_control = LoginControl()

    @commands.command(name='login')
    async def login(self, ctx, site: str):
        await ctx.send("Command recognized, passing data to control.")
```

```
        # Pass the command and site to control
        commandToPass = "login"
        result = await self.login_control.receive_command(commandToPass, site)

        # Send the result back to the user
        await ctx.send(result)



--- NavigationBoundary.py ---
from discord.ext import commands
from control.NavigationControl import NavigationControl

class NavigationBoundary(commands.Cog):

    def __init__(self):
        self.navigation_control = NavigationControl()                    # Initialize the control object

    @commands.command(name='navigate_to_website')
    async def navigate_to_website(self, ctx, url: str=None):
          await ctx.send("Command recognized, passing the data to control object.")        # Inform the
user that the command is recognized

        commandToPass = "navigate_to_website"
          result = self.navigation_control.receive_command(commandToPass, url)              # Pass the
command and URL to the control object
        await ctx.send(result)                                          # Send the result back to the user
```

```
--- PriceBoundary.py ---
from discord.ext import commands
from control.PriceControl import PriceControl

class PriceBoundary(commands.Cog):
    def __init__(self):
        # Initialize control objects directly
        self.price_control = PriceControl()

    @commands.command(name='get_price')
    async def get_price(self, ctx, url: str=None):
        """Command to get the price from the given URL."""
        await ctx.send("Command recognized, passing data to control.")
        # Pass the command to the control layer
        command_to_pass = "get_price"
        result = await self.price_control.receive_command(command_to_pass, url)
        await ctx.send(result)

    @commands.command(name='start_monitoring_price')
    async def start_monitoring_price(self, ctx, url: str = None, frequency: int = 20):
        """Command to monitor price at given frequency."""
            await ctx.send(f"Command recognized, starting price monitoring at {url} every {frequency}
second(s).")
        # Pass the command and data to the control layer
        command_to_pass = "monitor_price"
        response = await self.price_control.receive_command(command_to_pass, url, frequency)
```

```
        await ctx.send(response)

    @commands.command(name='stop_monitoring_price')
    async def stop_monitoring_price(self, ctx):
        """Command to stop monitoring the price."""
        await ctx.send("Command recognized, passing data to control.")
        # Pass the command to the control layer
        command_to_pass = "stop_monitoring_price"
        response = self.price_control.receive_command(command_to_pass)
        await ctx.send(response)



--- StopBoundary.py ---
from discord.ext import commands
from control.StopControl import StopControl

class StopBoundary(commands.Cog):
    def __init__(self):
        self.control = StopControl()  # Initialize control object

    @commands.command(name="stop_bot")
    async def stop_bot(self, ctx):
        await ctx.send("Command recognized, passing data to control.")

        # Pass the command to the control object
        commandToPass = "stop_bot"
        result = await self.control.receive_command(commandToPass, ctx)
```

print(result) # Send the result back to the Terminal. since the bot is shut down, it won't be able to send the message back to the user.

--- __init__.py ---

#empty init file