```
--- AccountBoundary.py ---

from discord.ext import commands

from control.AccountControl import AccountControl


class AccountBoundary(commands.Cog):

    def __init__(self, bot):

        self.bot = bot

        self.control = AccountControl()


    @commands.command(name="fetch_all_accounts")

    async def fetch_all_accounts(self, ctx):

        """Fetch all accounts from the database."""

        await ctx.send("Command recognized, taking action: Fetching all accounts.")

        accounts = self.control.fetch_all_accounts()

        if accounts:

                account_list = "\n".join([f"ID: {acc[0]}, Username: {acc[1]}, Password: {acc[2]}, Website: 
{acc[3]}" for acc in accounts])

            await ctx.send(f"Accounts:\n{account_list}")

        else:

            await ctx.send("No accounts found.")


    @commands.command(name="fetch_account_by_website")

    async def fetch_account_by_website(self, ctx, website: str):

        """Fetch an account by website."""

        await ctx.send(f"Command recognized, taking action: Fetching account for website {website}.")

        account = self.control.fetch_account_by_website(website)

        if account:
            await ctx.send(f"Account for {website}: Username: {account[0]}, Password: {account[1]}")
        else:
            await ctx.send(f"No account found for website {website}.")

    @commands.command(name="add_account")
    async def add_account(self, ctx, username: str, password: str, website: str):
        """Add a new account."""
        await ctx.send("Command recognized, taking action: Adding a new account.")
        result = self.control.add_account(username, password, website)
        if result:
            await ctx.send(f"Account for {website} added successfully.")
        else:
            await ctx.send(f"Failed to add account for {website}.")

    @commands.command(name="delete_account")
    async def delete_account(self, ctx, account_id: int):
        """Delete an account by ID."""
        await ctx.send(f"Command recognized, taking action: Deleting account with ID {account_id}.")
        result = self.control.delete_account(account_id)
        if result:
            await ctx.send(f"Account with ID {account_id} deleted successfully.")
        else:
            await ctx.send(f"Failed to delete account with ID {account_id}.")
```

--- CheckAvailabilityBoundary.py ---

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


class CheckAvailabilityBoundary(commands.Cog):
    def __init__(self, bot, browser_entity):
        self.bot = bot
        self.availibility_control = CheckAvailabilityControl(browser_entity)  # Initialize control object


    @commands.command(name="check_availability")
    async def check_availability(self, ctx, url: str, date_str=None):
        """Command to check availability at a given URL."""
        await ctx.send("Command recognized, taking action.")
        # Call the control layer to handle the availability check
        result = await self.availibility_control.check_availability(url, date_str)
        await ctx.send(result)



--- CloseBrowserBoundary.py ---
from discord.ext import commands
from control.CloseBrowserControl import CloseBrowserControl
from entity.BrowserEntity import BrowserEntity


class CloseBrowserBoundary(commands.Cog):
    def __init__(self, bot, browser_entity):
        self.bot = bot
         self.close_browser_control = CloseBrowserControl(browser_entity)  # Pass the browser_entity
to the control
```

```
    @commands.command(name='close_browser')
    async def close_browser(self, ctx):
        await ctx.send("Command recognized, taking action to close the browser.")
        result = self.close_browser_control.close_browser()
        await ctx.send(result)



--- GetPriceBoundary.py ---
from discord.ext import commands
from control.GetPriceControl import GetPriceControl


class GetPriceBoundary(commands.Cog):
    def __init__(self, bot, browser_entity):
        self.bot = bot
        self.price_control = GetPriceControl(browser_entity)

    @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, taking action.")
        response = await self.price_control.get_price(url)
        await ctx.send(response)



--- HelpBoundary.py ---
from discord.ext import commands
```

```python
from control.HelpControl import HelpControl


class HelpBoundary(commands.Cog):  # Cog to register with the bot
    def __init__(self, bot):
        self.bot = bot
        self.control = HelpControl()  # Initialize control object

    @commands.command(name="project_help")
    async def project_help(self, ctx):
        """Send a message with all the available commands."""
        await ctx.send("Command recognized, taking action.")
        response = self.control.get_help_message()
        await ctx.send(response)
```

--- LaunchBrowserBoundary.py ---

```python
from discord.ext import commands
from control.LaunchBrowserControl import LaunchBrowserControl


class LaunchBrowserBoundary(commands.Cog):
    def __init__(self, bot, browser_entity):
        self.bot = bot
        self.launch_browser_control = LaunchBrowserControl(browser_entity)  # Pass the browser_entity to the control

    @commands.command(name='launch_browser')
    async def launch_browser(self, ctx):
        await ctx.send("Command recognized, taking action.")
        result = self.launch_browser_control.launch_browser()
        await ctx.send(result)
```

```
--- LoginBoundary.py ---
from discord.ext import commands
from control.LoginControl import LoginControl

class LoginBoundary(commands.Cog):
    def __init__(self, bot, browser_entity):
        self.bot = bot
        self.login_control = LoginControl(browser_entity)  # Pass browser_entity to control

    @commands.command(name='login')
    async def login(self, ctx, site: str):
        await ctx.send("Command recognized, taking action.")
        result = await self.login_control.login(site)
        await ctx.send(result)
```

```
--- MonitorAvailabilityBoundary.py ---
from discord.ext import commands
from control.MonitorAvailabilityControl import MonitorAvailabilityControl

class MonitorAvailabilityBoundary(commands.Cog):
    def __init__(self, bot, monitor_availibility_control):
        self.bot = bot
        self.monitor_availibility_control = monitor_availibility_control  # Initialize control object

    @commands.command(name="monitor_availability")
    async def monitor_availability(self, ctx, url: str, date_str=None, frequency: int = 15):
        """Command to monitor availability at the given frequency."""
        await ctx.send("Command recognized, taking action.")
        await ctx.send(f"Monitoring availability at {url} every {frequency} second(s).")
            response = await self.monitor_availibility_control.start_monitoring_availability(url, date_str,
frequency)
        await ctx.send(response)

    @commands.command(name="stop_monitoring_availability")
    async def stop_monitoring(self, ctx):
        """Command to stop monitoring availability."""
        await ctx.send("Command recognized, taking action.")
        self.monitor_availibility_control.stop_monitoring()
        await ctx.send("Stopped monitoring availability.")



--- MonitorPriceBoundary.py ---
from discord.ext import commands
from control.MonitorPriceControl import MonitorPriceControl

class MonitorPriceBoundary(commands.Cog):
    def __init__(self, bot, monitor_price_control):
        self.bot = bot
        self.monitor_price_control = monitor_price_control  # Use shared instance


    @commands.command(name='start_monitoring_price')
    async def start_monitoring_price(self, ctx, url: str = None, frequency: int = 20):
            await ctx.send(f"Command recognized, starting price monitoring at {url} every {frequency}
second(s).")
        response = await self.monitor_price_control.start_monitoring_price(ctx, url, frequency)
        await ctx.send(response)



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


class NavigationBoundary(commands.Cog):
    def __init__(self, bot, browser_entity):
        self.bot = bot
        self.navigation_control = NavigationControl(browser_entity)


    @commands.command(name='navigate_to_website')
    async def navigate_to_website(self, ctx, url: str = None):
        await ctx.send("Command recognized, taking action.")
        result = self.navigation_control.navigate_to_website(url)
        await ctx.send(result)
```

--- StopBoundary.py ---

```python
from discord.ext import commands

from control.StopControl import StopControl


class StopBoundary(commands.Cog):

    def __init__(self, bot):

        self.bot = bot

        self.control = StopControl()


    @commands.command(name="stop_bot")

    async def stop_bot(self, ctx):

        """Shut down the bot."""

        await ctx.send("Command recognized, taking action")

        await self.control.stop_bot(ctx, self.bot)  # Call the control's method to stop the bot
```

--- StopMonitoringPriceBoundary.py ---

```python
from discord.ext import commands

from control.MonitorPriceControl import MonitorPriceControl


class StopMonitoringPriceBoundary(commands.Cog):

    def __init__(self, bot, monitor_price_control):

        self.bot = bot

        self.monitor_price_control = monitor_price_control  # Use shared instance


    @commands.command(name='stop_monitoring_price')

    async def StopMonitoringPrice(self, ctx):
        """Command to stop monitoring the price."""

        await ctx.send("Command recognized, taking action.")

        response = self.monitor_price_control.stop_monitoring()

        await ctx.send(response)




--- __init__.py ---

#empty init file
```