```python
from datetime import datetime
from os.path import exists

#   Initializing TODOs as an empty array
TODOS = []
VALID_DATE_FORMAT = "yyyy-mm-dd"
VALID_TIME_FORMAT = "hh:mm"


#   A function to make redundant task easier
def invalid_error():
    print("Invalid Input")


#   A function used for taking user confirmation
def is_confirmed(text):
    decision = input(f"{text} (y,n) : ")
    if decision.lower() == "y":
        return True
    return False


#   Function to get valid date of a month based on the month and year
def valid_date_by_month(year):
    return {
        1: 31,
        2: 29 if (int(year) % 4 == 0) else 28,
        3: 31,
        4: 30,
        5: 31,
        6: 30,
        7: 31,
        8: 31,
        9: 30,
        10: 31,
        11: 30,
        12: 31,
    }


#   Function to check if a string is a valid date
def is_valid_date(str):
    if str == "0":
        return False
    if len(str.split("-")) != 3:
        invalid_error()
        return False
    for part in str.split("-"):
        if not part.isdigit():
            invalid_error()
            return False
    year, month, day = [int(x) for x in str.split("-")]
    if (
        year in range(1970, 3000)
        and month in range(1, 13)
        and (day in range(1, 1 + valid_date_by_month(year)[month]))
    ):
        return True
    invalid_error()
    return False
```

```python
#   Function to check if a string is a valid time
def is_valid_time(str):
    if str == "0":
        return False
    parts = str.split(":")
    if len(parts) != 2:
        invalid_error()
        return False
    for part in parts:
        if not part.isdigit():
            invalid_error()
            return False
    hour, minute = [int(x) for x in str.split(":")]
    if hour in range(0, 24) and minute in range(0, 60):
        return True
    invalid_error()
    return False


#   Function to get datetime object from a date string and time string
def get_datetime(dateStr, timeStr):
    if is_valid_date(dateStr) and is_valid_time(timeStr):
        year, month, day = [int(x) for x in dateStr.split("-")]
        hour, minute = [int(x) for x in timeStr.split(":")]
        return datetime(year, month, day, hour, minute)
    return None


#   Function to check if it has any time clash with existing TODOs
def has_time_clash(time):
    for item in TODOS:
        if int(time.timestamp() * 1000) in range(
            int(item["start_time"] * 1000), int(item["end_time"] * 1000) + 1
        ):
            return True
    return False


#   Function to take user input on adding and updating TODOs
def take_todo_input(return_func):
    description = input("Description : ")
    startDateStr = "0"
    while not is_valid_date(startDateStr):
        startDateStr = input(f"Start Date ({VALID_DATE_FORMAT}) : ")
    startTimeStr = "0"
    while not is_valid_time(startTimeStr):
        startTimeStr = input(f"Start Time ({VALID_TIME_FORMAT}) : ")
    if has_time_clash(get_datetime(startDateStr, startTimeStr)):
        print("You have another todo at the same time, starting over...")
        return_func()
        return None
    endDateStr = "0"
    while not is_valid_date(endDateStr):
        endDateStr = input(f"End Date ({VALID_DATE_FORMAT}) : ")
    endTimeStr = "0"
    while not is_valid_time(endTimeStr):
        endTimeStr = input(f"End Time ({VALID_TIME_FORMAT}) : ")
    if has_time_clash(get_datetime(endDateStr, endTimeStr)):
        print("You have another todo at the same time, starting over...")
        return_func()
        return None
    place = input("Place : ")
```

```
    return {
        "description": description,
        "place": place,
        "start_time": get_datetime(startDateStr, startTimeStr).timestamp(),
        "end_time": get_datetime(endDateStr, endTimeStr).timestamp(),
    }


#   Function for adding new todo item
def add_todo():
    print("New Todo : ")
    todo = take_todo_input(add_todo)
    if todo:
        TODOS.append(todo)
        print_all_todos()
        if is_confirmed("Add again?"):
            add_todo()


#   Function for printing out args todos
def print_todos(todos):
    if len(todos) > 0:
        index = 0
        for item in todos:
            index += 1
            print("\n", index, ". ")
            print(item["description"])
            print("Start : ", datetime.fromtimestamp(item["start_time"]))
            print("End : ", datetime.fromtimestamp(item["end_time"]))
            print("Place : ", item["place"], "\n")


#   Function for printing out all todos to the console
def print_all_todos():
    print_todos(TODOS)
    if len(TODOS) == 0:
        if is_confirmed("No TODOs added, Add one?"):
            add_todo()


#   Function for updating an existing todo item
def update_todo():
    print_all_todos()
    option = int(input("Enter a number to update : "))
    if option <= len(TODOS):
        todo = take_todo_input(update_todo)
        if todo:
            TODOS[option - 1] = todo
            print_all_todos()
            print("Updated TODO")
    else:
        print("Invalid input")


#   Function for removing an existing todo item
def remove_todo():
    print_all_todos()
    option = int(input("Enter a number to remove : "))
    if option <= len(TODOS):
        del TODOS[option - 1]
        print_all_todos()
        if is_confirmed("Delete again?"):
```

```
            remove_todo()
    else:
        print("Invalid input")


#   Function to print all upcoming task for today
def today_tasks():
    now = datetime.now()
    end_of_today = datetime(now.year, now.month, now.day, 23, 59)
    print("Upcoming tasks for today : ")
    print_todos(
        [
            todo
            for todo in TODOS
            if int(todo["start_time"] * 1000) > int(now.timestamp() * 1000)
            and int(todo["start_time"] * 1000) < int(end_of_today.timestamp() *
1000)
        ]
    )


#   Function to initialize global variable TODO with todo items from data.txt
def init():
    if exists("ToDo/data.txt"):
        data_file = open("ToDo/data.txt", "r")
        data = data_file.readlines()
        for line in data:
            if len(line.split("|")) == 4:
                description, place, start_time, end_time =
line.strip().split("|")
                TODOS.append(
                    {
                        "description": description,
                        "place": place,
                        "start_time": float(start_time),
                        "end_time": float(end_time),
                    }
                )
        data_file.close()


#   Function to save all todo items to data.txt file
def save_file():
    data_file = open("ToDo/data.txt", "w")
    data_file.writelines(
        f'{("|".join((str(item[key]) for key in item)))}\n' for item in TODOS
    )
    data_file.close()


#   Main function containing the menu of application
def main():
    init()
    while True:
        print(
            """
        Welcome to TodoList

        1. View TODOs
        2. Upcoming TODOs Today
        3. Add New TODO
        4. Update TODO
```

```
        5. Remove TODO
        0. Exit
        """
        )
        option = int(input("Enter an option : "))
        if option == 0:
            break
        menus = {
            1: print_all_todos,
            2: today_tasks,
            3: add_todo,
            4: update_todo,
            5: remove_todo,
        }
        if menus.keys().__contains__(option):
            menus[option]()
        else:
            print("Invalid Option")
    save_file()


#   Executing main function to start execution
main()
```