POMODORO
BASED PRODUCTIVITY APP

TEAM MEMBER
SAI, CAPPI & KHOI

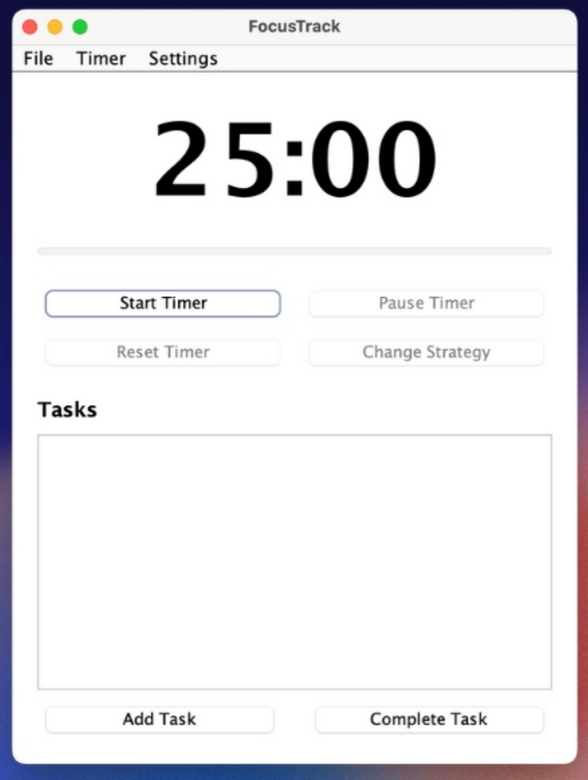
Tech Stack
Java, Java Swing

FOCUS TRACK

TEAM MEMBER
SAI, CAPPI & KHOI

Tech Stack
Java, Java Swing

FOCUS TRACK
PRODUCTIVITY APP




```
===== Task Timer Application =====
1. Add Task
2. List Tasks
3. Start Timer for Task
4. Pause Timer
5. Resume Timer
6. Cancel Timer
7. Mark Task as Completed
8. Sound Settings
9. Exit

Sound: ON (Volume: 100%)
Choose an option:
```

PURPOSE

HELP STUDENTS AND PROFESSIONALS
MANAGE TIME, TRACK TASKS, & BOOST PRODUCTIVITY
USING SCIENTIFICALLY-BACKED FOCUS METHODS

USE CASE SCENARIOS

Target User

Problem

Solution

USE CASE SCENARIOS

1. Target User

Student

Freelancer

Remote Worker

2. Real-world Problems

Difficulty staying focused for long periods

Struggles with organizing tasks and prioritizing work

Lack of clear feedback or progress tracking in daily routines

Getting distracted during long work sessions

SOLUTION

& Feature of FOCUS TRACK

SOLUTION

& Feature of FOCUS TRACK

S1 : Pomodoro Technique Integration

Encourages focused work intervals (e.g., 25 mins) followed by short breaks (5 mins)

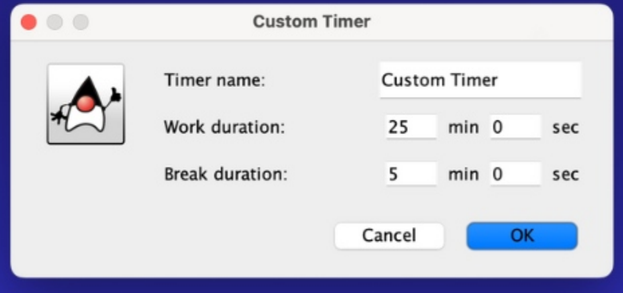
Reduces burnout and mental fatigue

Timer automatically transitions between work and rest phases

S2 : Flexible Timer Modes

Supports Pomodoro, Short Break, Long Break, and Custom strategies

Easily switchable depending on the user's needs or energy level



SOLUTION

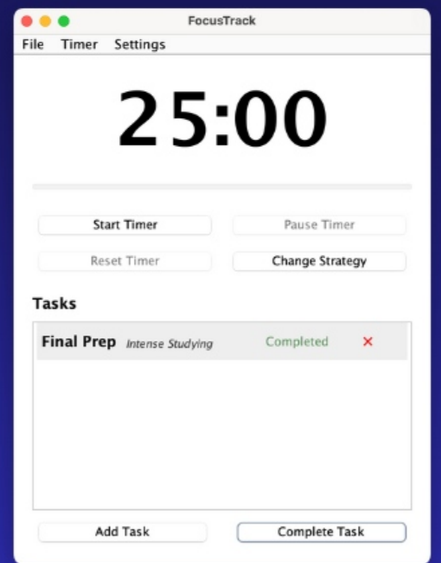
& Feature of FOCUS TRACK

S3 : Task Management System

Add, update, and track tasks

Tasks are linked to Pomodoro sessions

Each task tracks how many Pomodoros were completed



S4 : Productivity Metrics

Tracks completed focus sessions per task

Users can visually see how much progress they've made

Encourages consistency and builds motivation

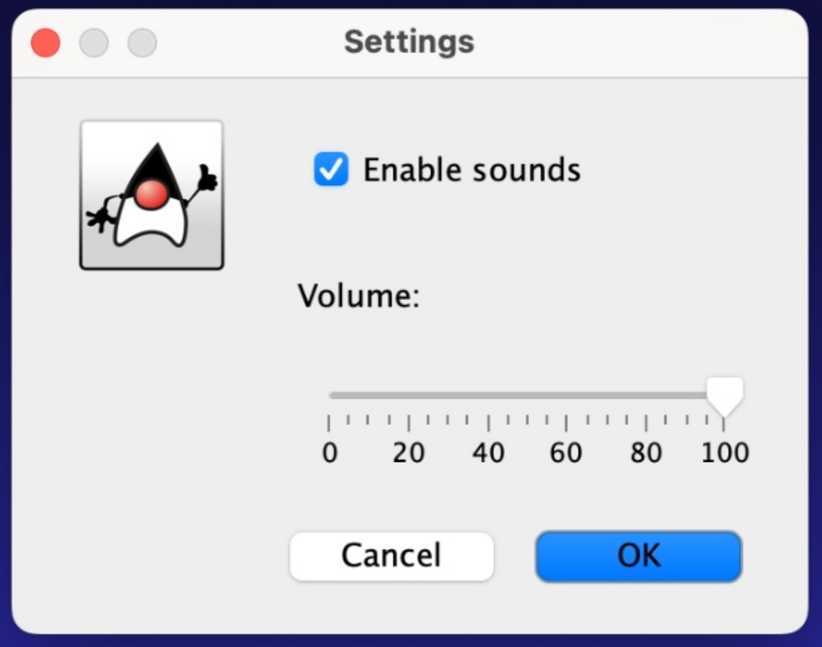
EXTRA FEATURE

Smart Sound Notifications

Distinct Sounds for Each Phase
Focus Session Finished & Break Over

Helps users recognize the timer phase without looking at the screen

Reduces context switching
— stay in flow, just follow the sound



FOCUS TRACK

PRODUCTIVITY APP

ARCHITECTURE & OOP PRINCIPLE

ARCHITECTURE & OOP PRINCIPLE

Modular Package Design

Five Key Packages

app – Application entry point (FocusTrackLauncher)

model – Core logic: Task, TaskManager, TaskStatus

timer – Timer logic with Strategy Pattern support

ui – Swing-based UI components and listeners

io – File read/write using TaskFileHandler

```
FocusTrack/
├── src/
│   ├── main/
│   │   ├── app/
│   │   │   └── App.java
│   │   ├── io/
│   │   │   └── TaskFileHandler.java
│   │   │
│   │   ├── model/
│   │   │   ├── Task.java
│   │   │   ├── TaskManager.java
│   │   │   └── Timer.java
│   │   │
│   │   ├── resources/
│   │   │   ├── sounds/
│   │   │   │   ├── break_complete.wav
│   │   │   │   └── error.wav
│   │   │   │   └── timer_complete.wav
│   │   │   └── tasks.txt/
│   │   │
│   │   ├── timer/
│   │   │   ├── LongBreakTimer.java
│   │   │   ├── PomodoroTimer.java
│   │   │   ├── ShortBreakTimer.java
│   │   │   ├── TaskTimer.java
│   │   │   ├── TimerManager.java
│   │   │   ├── TimerStrategy.java
│   │   │   └── TimerStrategyFactory.java
│   │   │
│   │   ├── ui/
│   │   │   ├── FocusTrackUI.java
│   │   │   ├── TimerDisplayManager.java
│   │   │   └── TimerUI.java
│   │   └── FocusTrackLauncher.java
└── README.md
```

ARCHITECTURE & OOP PRINCIPLE

OOP Principles Applied

1. Encapsulation

Helps protect data integrity and control object access

Implementation examples:

Task.java :

Private fields with public getters/setters

TaskTimer.java :

Internal state managed through controlled methods

```java
// From Task.java
private String title;
private String description;
private TaskStatus status;
private int completedPomodoros;

public String getTitle() {
    return title;
}

public void setTitle(String title) {
    this.title = title;
}
```

ARCHITECTURE & OOP PRINCIPLE

OOP Principles Applied

2. Inheritance

Supports code reuse and polymorphism through interface-based inheritance.

Implementation examples:

TimerStrategy :

implemented by PomodoroTimer, ShortBreakTimer, etc.

3. Abstraction

Simplifies complex processes and improves code readability.

Implementation examples:

TimerStrategy :

interface abstracts timing details

TaskTimer :

abstracts timer execution logic

```java
package main.timer;

public interface TimerStrategy {
    int getWorkDuration();
    int getBreakDuration();
    String getName();
    String getDescription();

    // Default implementations for backward compatibility
    default int getWorkDurationSeconds() {
        return 0;
    }

    default int getBreakDurationSeconds() {
        return 0;
    }

    // Helper methods to get total seconds
    default int getTotalWorkSeconds() {
        return getWorkDuration() * 60 + getWorkDurationSeconds();
    }

    default int getTotalBreakSeconds() {
        return getBreakDuration() * 60 + getBreakDurationSeconds();
    }
}
```

ARCHITECTURE & OOP PRINCIPLE

OOP Principles Applied

4. Polymorphism

Enables flexible and extensible code behavior

Implementation examples:

TimerStrategy :

interface with multiple implementations

***TaskTimer.changeStrategy()**:*

Usage of method in TaskTimer.java

```
// TaskTimer can work with any TimerStrategy implementation
public TaskTimer(Task task, TimerStrategy strategy) {
    this.task = task;
    this.strategy = strategy;
    // Other initialization...
}
```

FOCUS TRACK
PRODUCTIVITY APP

DESIGN PATTERN
IMPLEMENTATION ANALYSIS

DESIGN PATTERN
IMPLEMENTATION ANALYSIS

1. Singleton Pattern

Ensures only one instance of the task manager and timer manager exists across the app. Maintains a centralized, consistent state for tasks and timers.

Implementation examples:

TaskManager.java : *(lines 5-19)*

TimerManager.java : *(lines 12-22)*

```java
// From TaskManager.java
private static TaskManager instance = null;

private TaskManager() {
    fileHandler = new TaskFileHandler();
    autoSave = true;
    loadTasks();
}

public static TaskManager getInstance() {
    if (instance == null) {
        instance = new TaskManager();
    }
    return instance;
}
```

DESIGN PATTERN
IMPLEMENTATION ANALYSIS

2. Strategy Pattern

Allows switching between

different timer behaviors at runtime.

Supports multiple productivity techniques

(Pomodoro, Custom, etc).

Implementation examples:

TimerStrategy.java (interface)

PomodoroTimer.java, ShortBreakTimer.java,

LongBreakTimer.java (concrete implementations)

TaskTimer.java (context class using strategies)

```java
// TimerStrategy interface
public interface TimerStrategy {
    int getWorkDuration();
    int getBreakDuration();
    String getName();
    String getDescription();
    // Default implementations...
}

// PomodoroTimer implementation
public class PomodoroTimer implements TimerStrategy {
    @Override
    public int getWorkDuration() {
        return 25;
    }

    @Override
    public int getBreakDuration() {
        return 5;
    }
    // Other implementations...
}

// In TaskTimer.java
public void changeStrategy(TimerStrategy newStrategy) {
    boolean wasRunning = isRunning;

    if (wasRunning) {
        pause();
    }

    this.strategy = newStrategy;
    this.isWorkPhase = true;
    this.remainingSeconds = newStrategy.getTotalWorkSeconds
();

    if (wasRunning) {
        resume();
    }
}
```

DESIGN PATTERN
IMPLEMENTATION ANALYSIS

3. Factory Method Pattern

Centralizes creation of timer strategies.

Simplifies how standard or

custom timer types are built.

Implementation examples:

TimerStrategyFactory.java

```
public class TimerStrategyFactory {
    public static TimerStrategy createStrategy(TimerStrategy
Type strategyType) {
        switch (strategyType) {
            case POMODORO:
                return new PomodoroTimer();
            case SHORT_BREAK:
                return new ShortBreakTimer();
            case LONG_BREAK:
                return new LongBreakTimer();
            default:
                // Default to Pomodoro
                return new PomodoroTimer();
        }
    }

    public static TimerStrategy createCustomStrategy(
        final String name,
        final String description,
        final int workMinutes,
        final int workSeconds,
        final int breakMinutes,
        final int breakSeconds) {
        // Implementation...
    }
}
```

DESIGN PATTERN
IMPLEMENTATION ANALYSIS

4. Observer Pattern

Notifies the UI when timer states change
(tick, complete, switch phases).
Supports real-time updates in the interface.

Implementation examples:

TaskTimer.java

(lines 16-21)

TimerListener interface

```
public interface TimerListener {
    void onTick(int seconds);
    void onPhaseComplete(boolean wasWorkPhase);
    void onTimerComplete();
}

public void setListener(TimerListener listener) {
    this.listener = listener;
}
```

DESIGN PATTERN
IMPLEMENTATION ANALYSIS

4. Observer Pattern

Notifies the UI when timer states change (tick, complete, switch phases).

Supports real-time updates in the interface.

Implementation examples:

TaskTimer.java

(lines 16-21)

TimerListener interface

```
public interface TimerListener {
    void onTick(int seconds);
    void onPhaseComplete(boolean wasWorkPhase);
    void onTimerComplete();
}

public void setListener(TimerListener listener) {
    this.listener = listener;
}
```

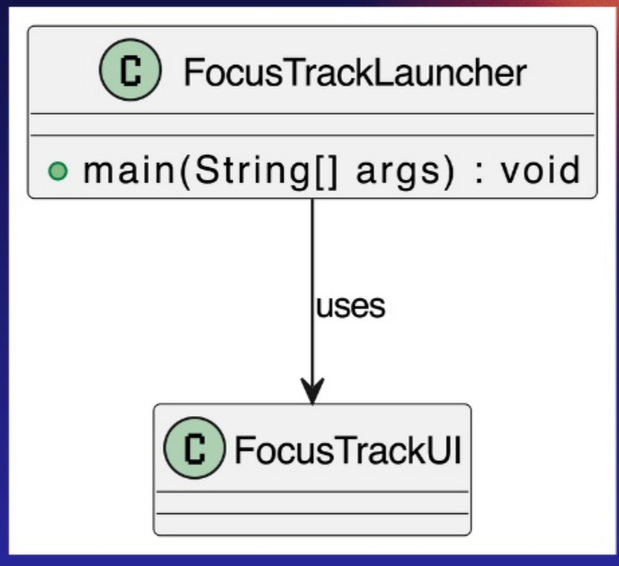
FOCUS TRACK
PRODUCTIVITY APP

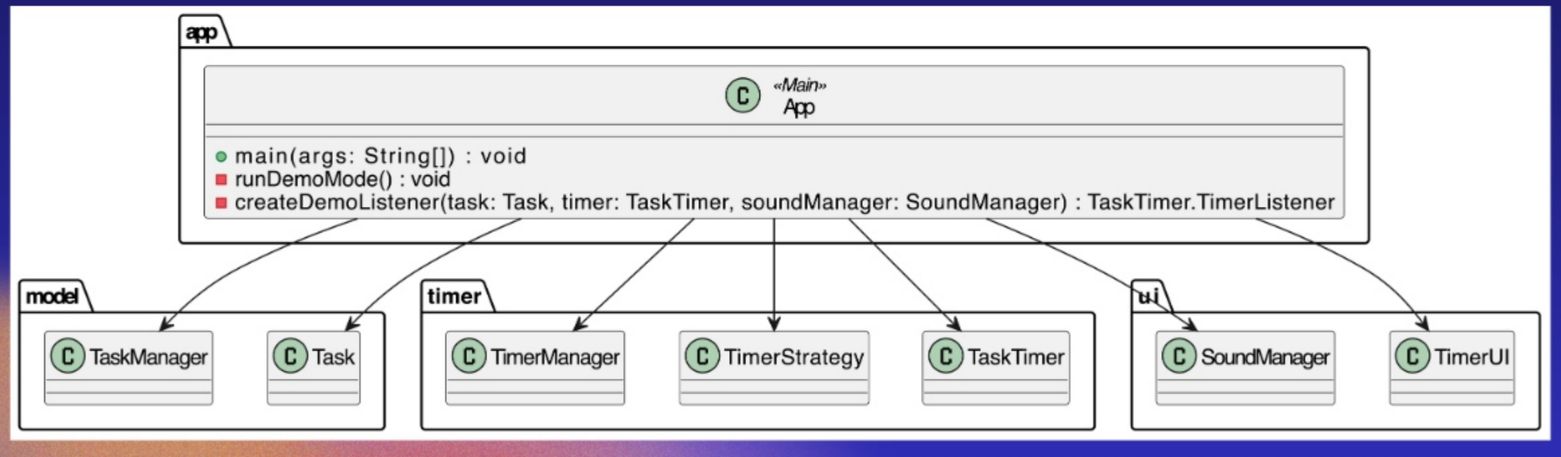
CLASS DIAGRAM
UML DIAGRAM

CLASS DIAGRAM
UML DIAGRAM

APP & Launcher

Entry point of the application.
Handles startup and demo mode logic.





CLASS DIAGRAM
UML DIAGRAM

IO Package

Handles file input/output
(saving and loading tasks).

Key Class:

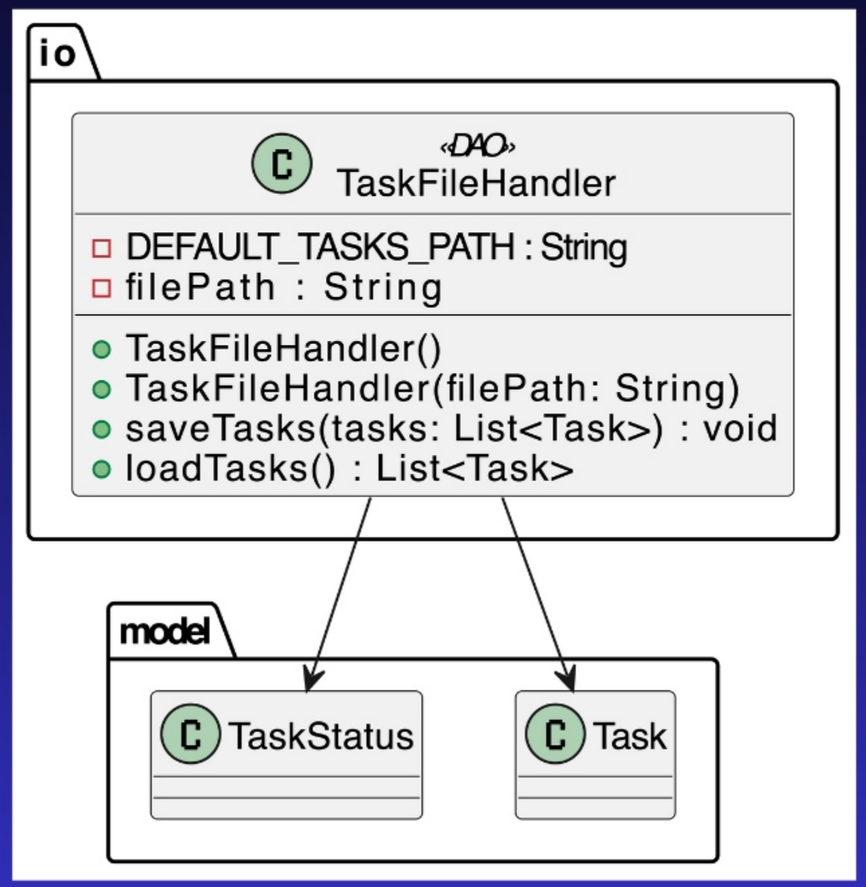
TaskFileHandler <<DAO>>:

Reads from and writes to tasks.txt

Used By:

TaskManager for persistence

Depends on Task, TaskStatus



CLASS DIAGRAM
UML DIAGRAM

Model Package

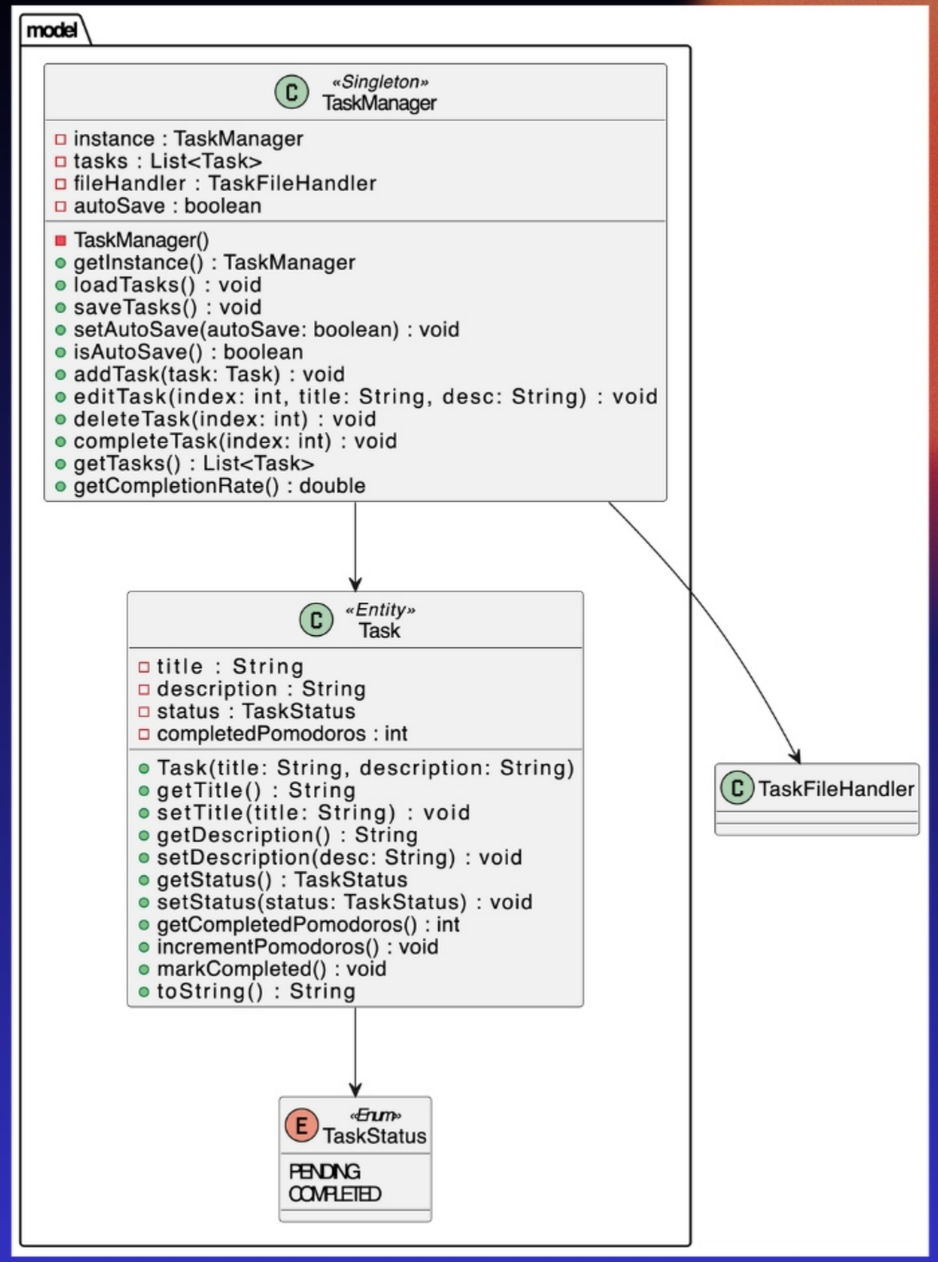
Core task and
task management logic.

Relationship:

TaskManager uses
TaskFileHandler (from io)
Task uses TaskStatus

Key Class:

Task <<Entity>>:
Represents a single task with title,
description, status, and Pomodoro count.
TaskManager <<Singleton>>: Manages a list of Tasks.
TaskStatus <<Enum>>:
Defines task states (PENDING, COMPLETED)



CLASS DIAGRAM
UML DIAGRAM

Timer Package

Provides the Pomodoro
timer system and timer strategies.

Key Class:

TimerStrategy <<interface>>:

Strategy for different timer types.

PomodoroTimer, ShortBreakTimer,

LongBreakTimer <<Class>>:

Implement TimerStrategy.

TaskTimer <<Class>>:

Manages an active timer for a task.

Relationship:

TaskTimer uses Task and TimerStrategy

TimerManager holds a map of Task
→ TaskTimer

App and UI use this package heavily

TimerManager <<Singleton>>:

Manages all active TaskTimers.

TimerStrategyFactory <<Factory>>:

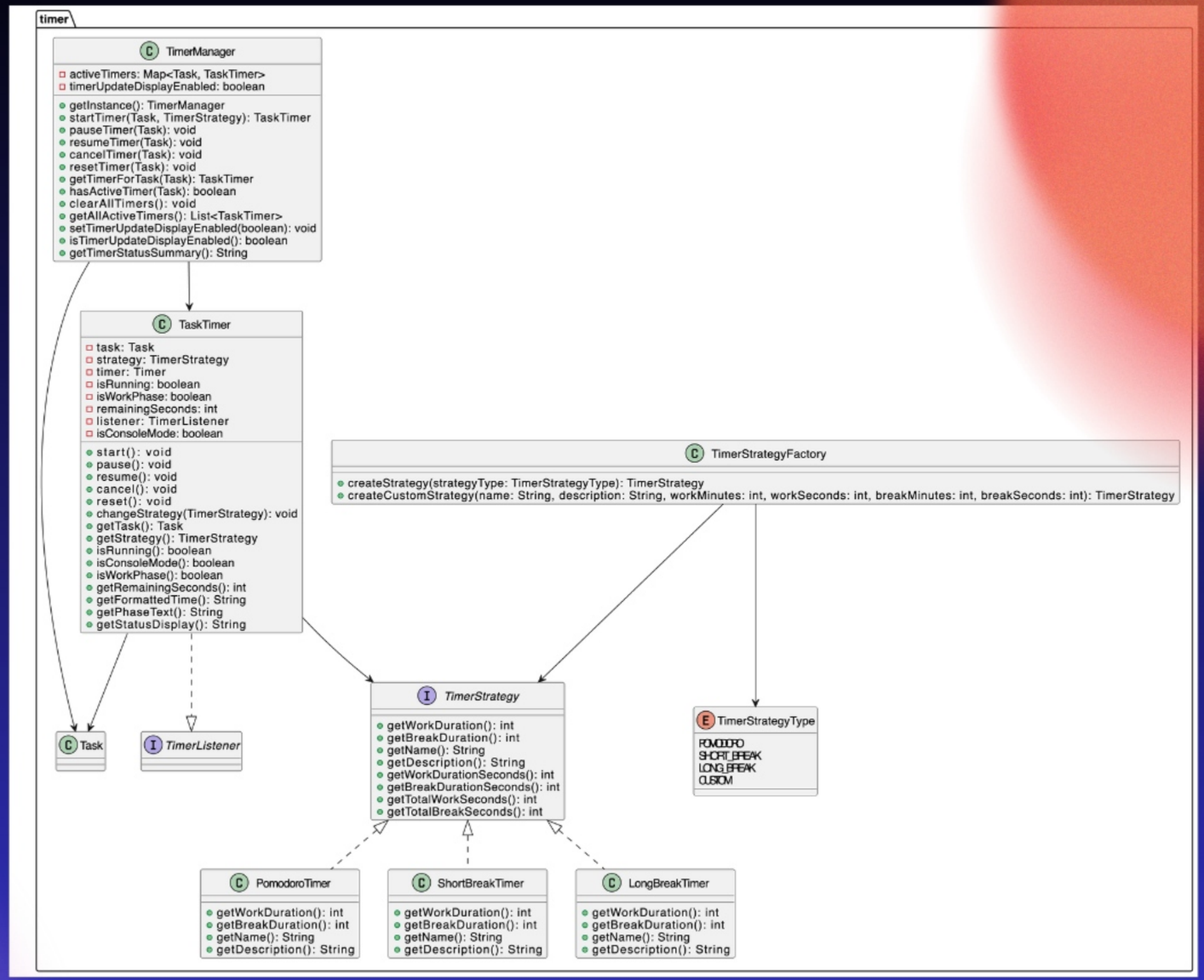
Creates timer strategies.

TimerStrategyType <<Enum>>:

Enum values used for strategy selection.

CLASS DIAGRAM
UML DIAGRAM

Timer Package



CLASS DIAGRAM
UML DIAGRAM

UI Package

Java Swing GUI and user interaction.

Key Class:

FocusTrackUI <<UI Component>>:

Main app window with timer and task control.

AddTaskDialog, TimerStrategyDialog,

CustomTimerDialog, SettingsDialog, TaskListPanel

<<UI Components>>:

Modal dialogs and panels for task/timer configuration.

TimerDisplayManager:

Console-based timer visualization.

TimerUI: Alternate timer view with its own controls.

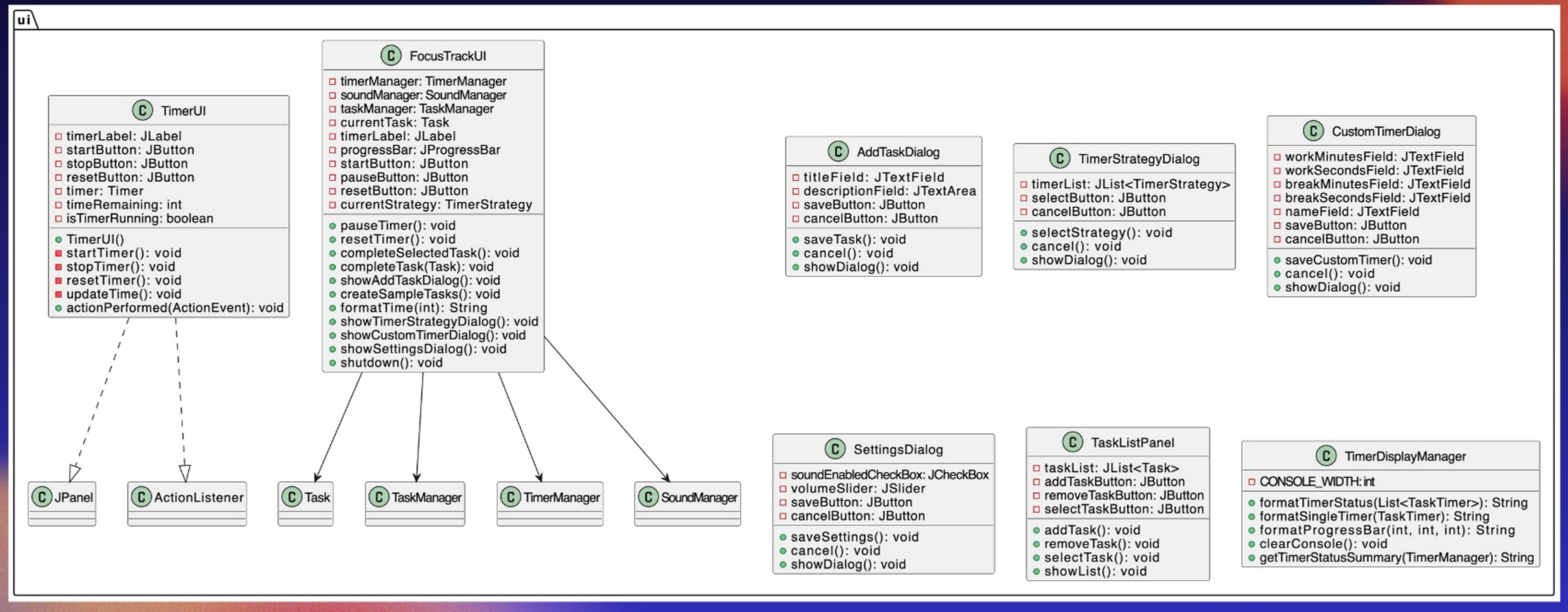
Relationship:

Uses Task, TaskManager, TaskTimer,

TimerStrategy, SoundManager, TimerManager

CLASS DIAGRAM UI Package

UML DIAGRAM



FOCUS TRACK

PRODUCTIVITY APP

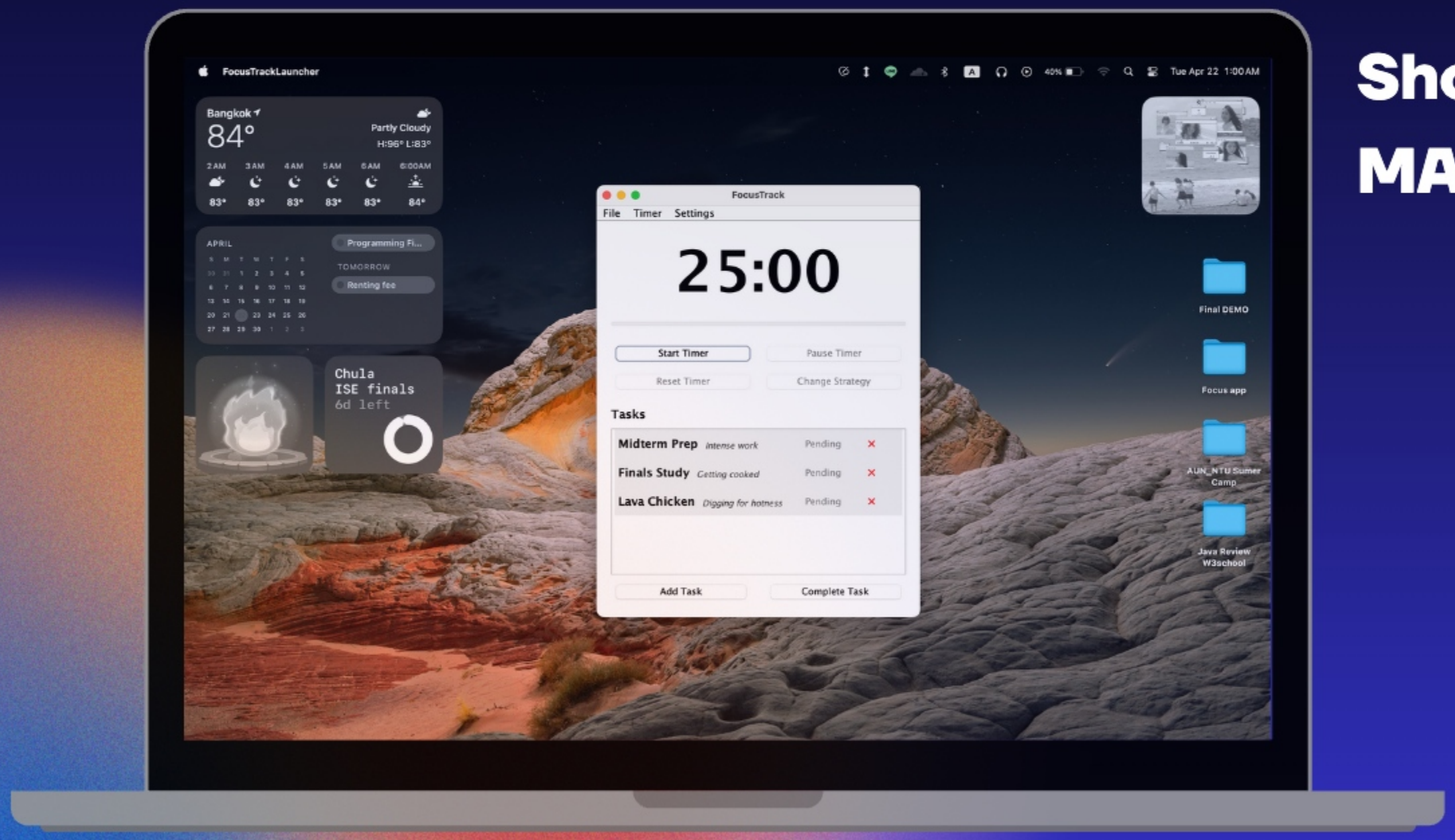
LIVE DEMOSTRATION | 10

TEAM MEMBER
SAI, CAPPI & KHOI

Tech Stack
Java, Java Swing

FOCUS TRACK

PRODUCTIVITY APP



Showcasing

MAIN DASHBOARD

LIVE

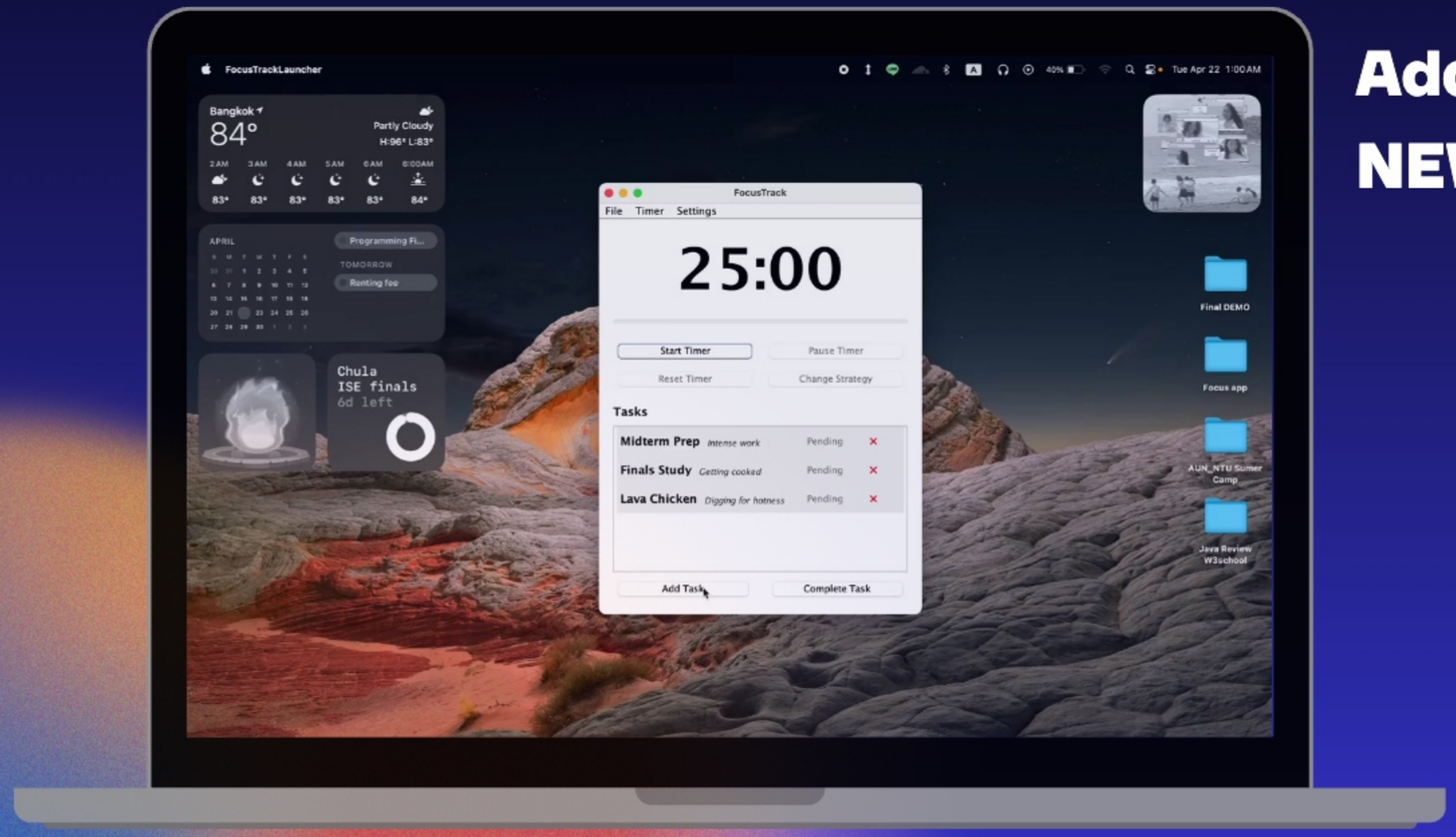
DEMOSTRATION

TEAM MEMBER
SAI, CAPPI & KHOI

Tech Stack
Java, Java Swing

FOCUS TRACK

PRODUCTIVITY APP



Adding NEW TASKS

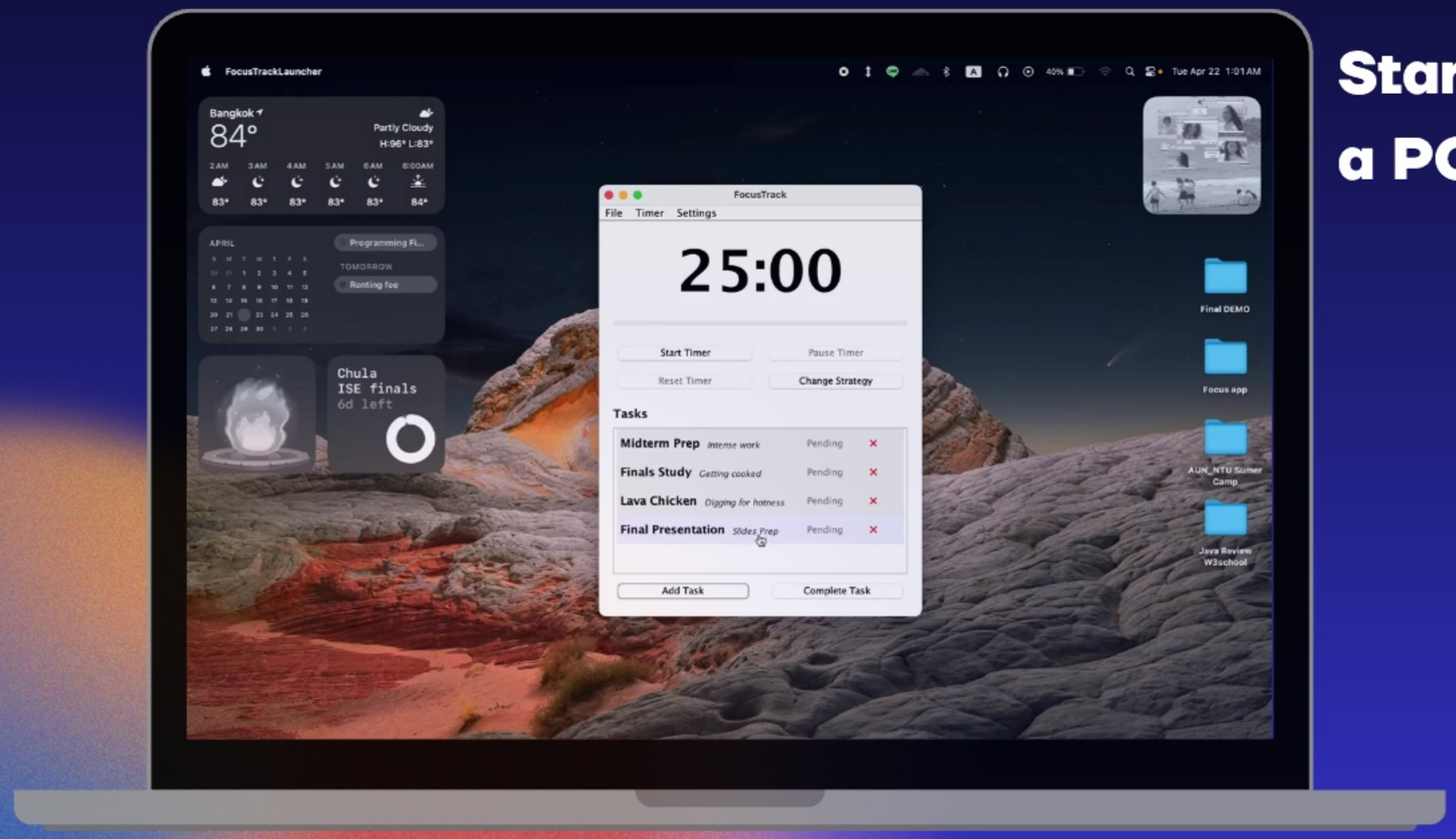
LIVE DEMOSTRATION

TEAM MEMBER
SAI, CAPPI & KHOI

Tech Stack
Java, Java Swing

FOCUS TRACK

PRODUCTIVITY APP



Starting
a POMODORO Session

LIVE
DEMOSTRATION

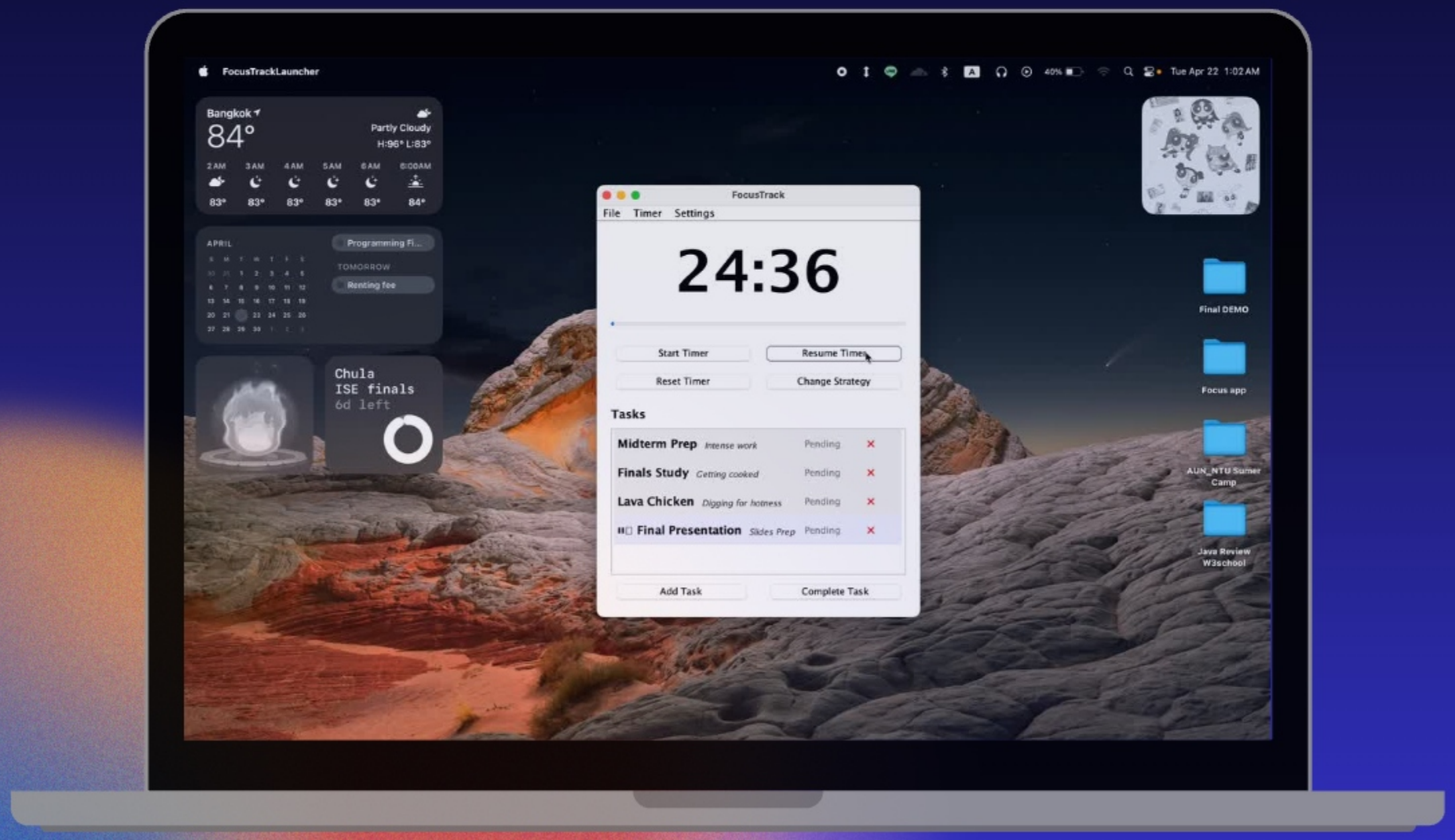
TEAM MEMBER
SAI, CAPPI & KHOI

Tech Stack
Java, Java Swing

FOCUS TRACK

PRODUCTIVITY APP

PAUSING Timer



LIVE

DEMOSTRATION

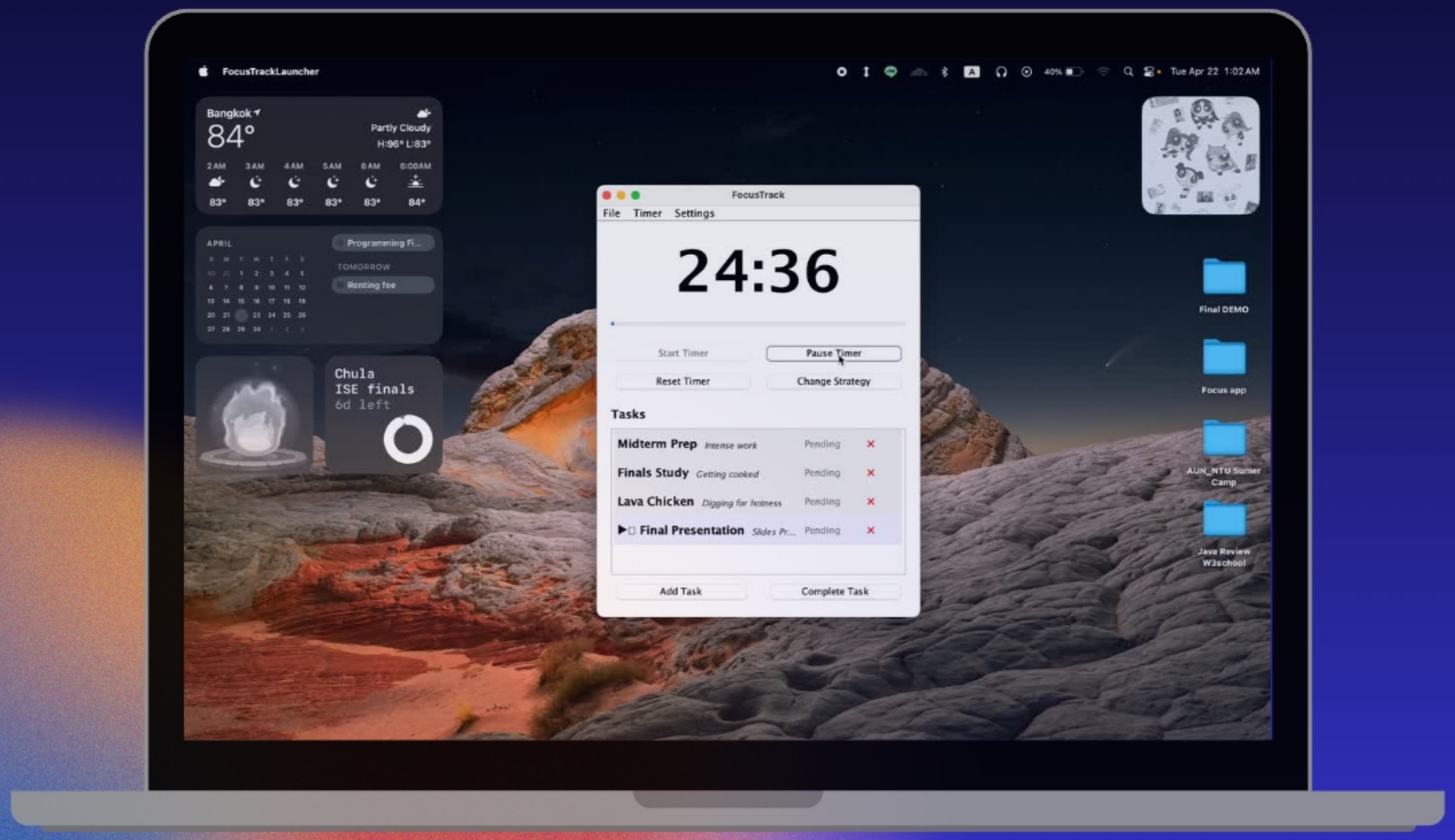
TEAM MEMBER
SAI, CAPPI & KHOI

Tech Stack
Java, Java Swing

FOCUS TRACK

PRODUCTIVITY APP

RESUMING Timer



LIVE DEMOSTRATION

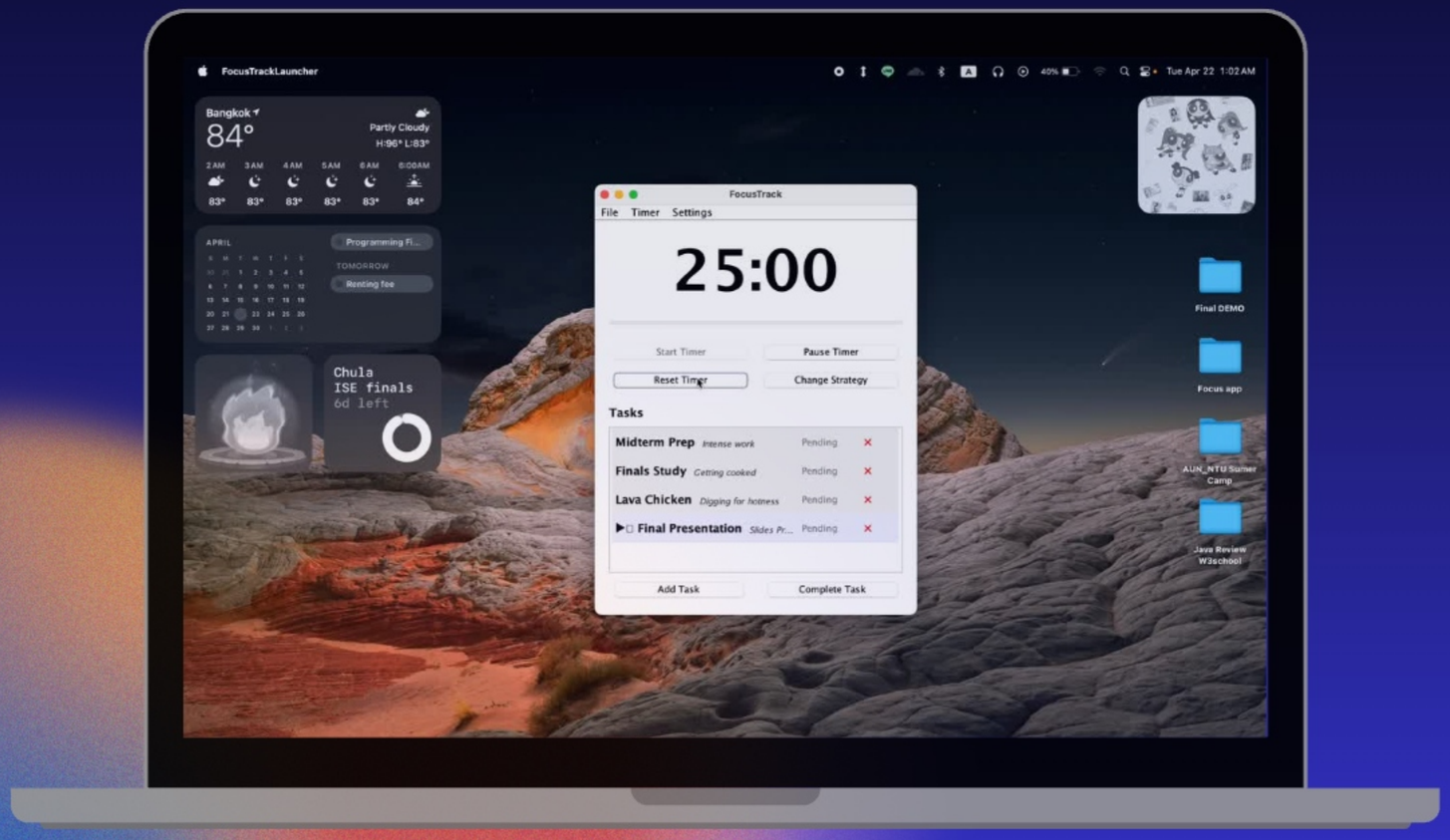
TEAM MEMBER
SAI, CAPPI & KHOI

Tech Stack
Java, Java Swing

FOCUS TRACK

PRODUCTIVITY APP

RESETING Timer



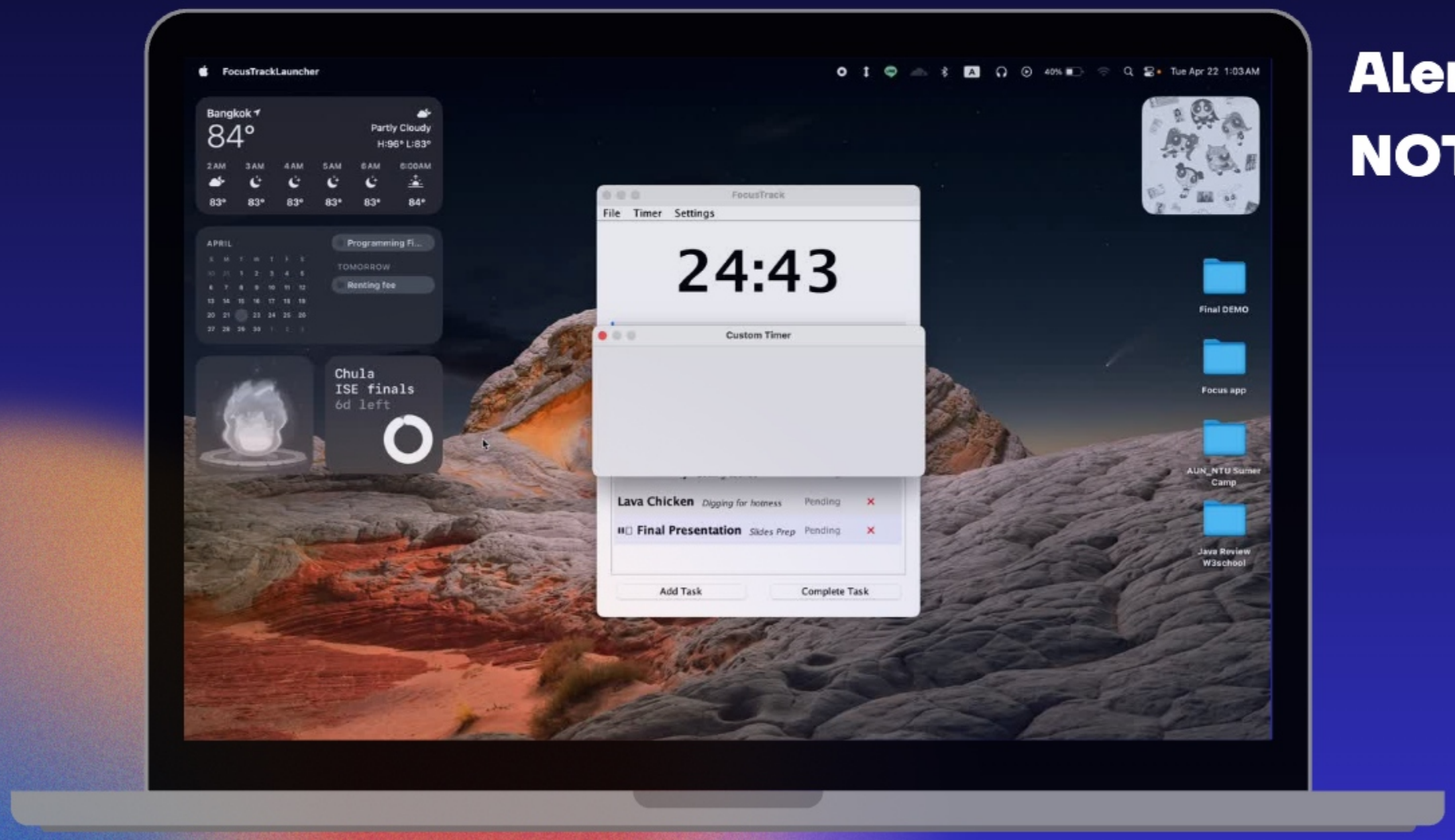
LIVE

DEMOSTRATION

TEAM MEMBER
SAI, CAPPI & KHOI

Tech Stack
Java, Java Swing

FOCUS TRACK
PRODUCTIVITY APP



Alert

NOTIFICATION

LIVE
DEMOSTRATION

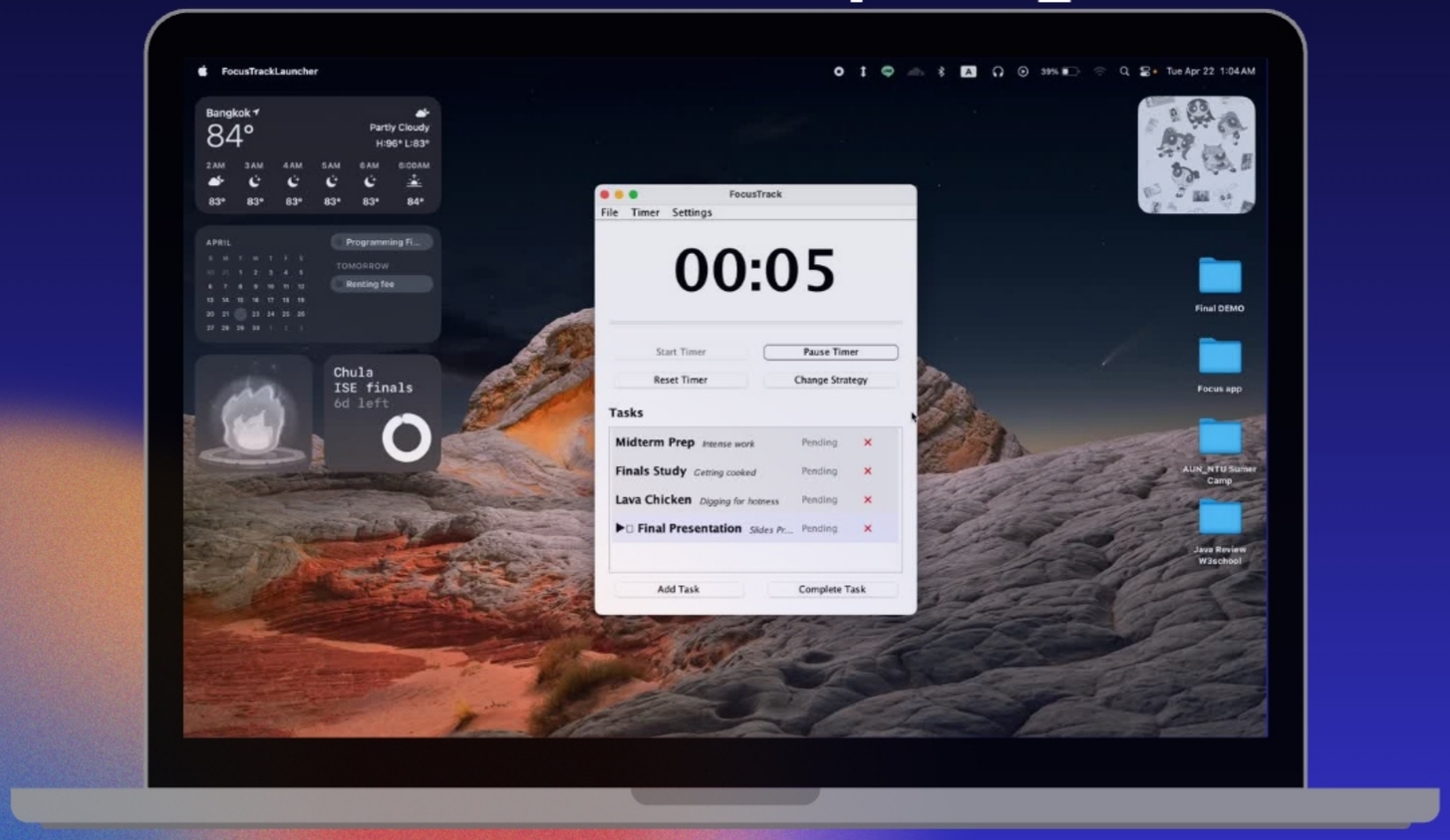
TEAM MEMBER
SAI, CAPPI & KHOI

Tech Stack
Java, Java Swing

FOCUS TRACK

PRODUCTIVITY APP

Completing TASK



LIVE

DEMOSTRATION

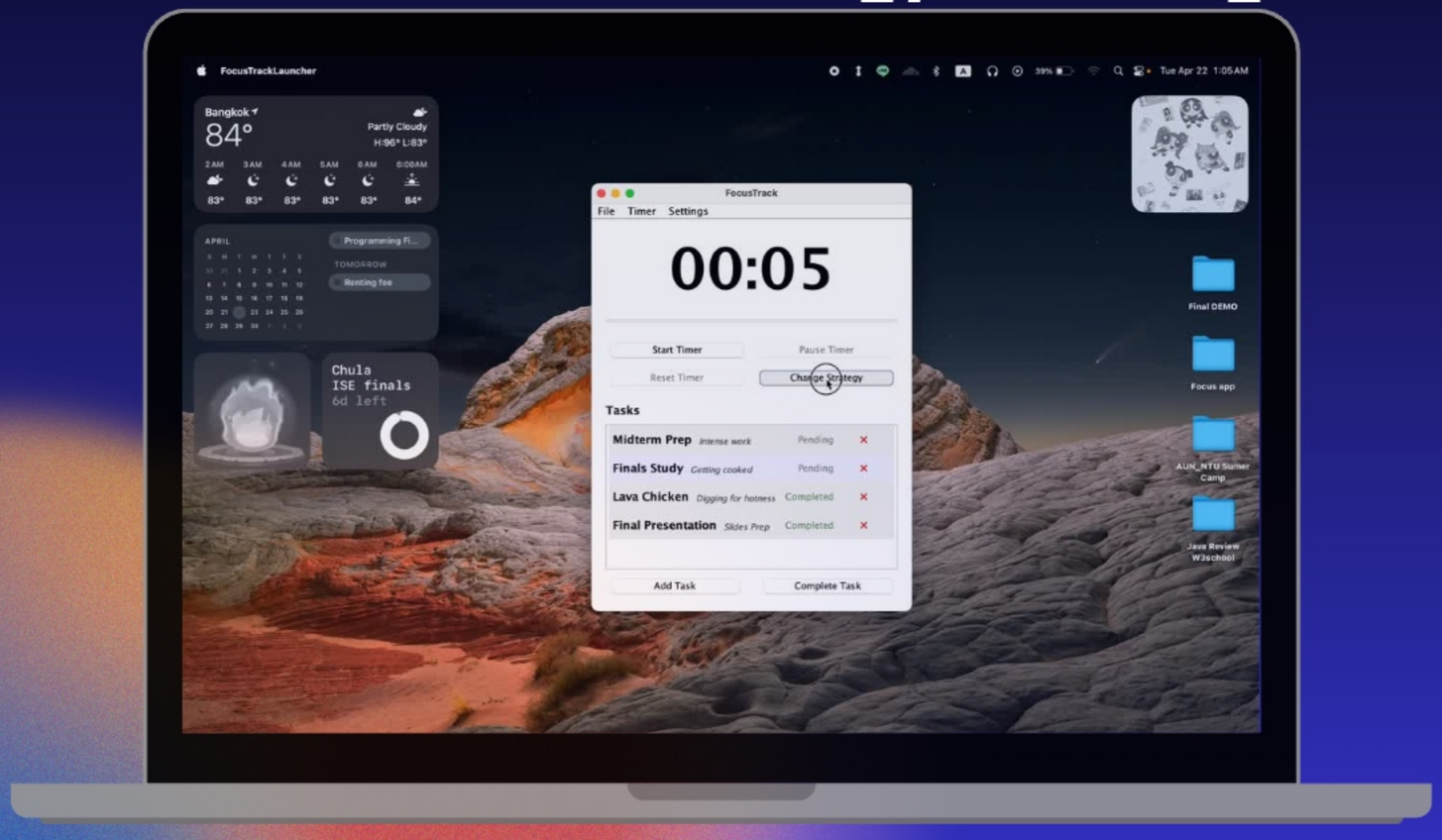
TEAM MEMBER
SAI, CAPPI & KHOI

Tech Stack
Java, Java Swing

FOCUS TRACK

PRODUCTIVITY APP

Strategy Switching



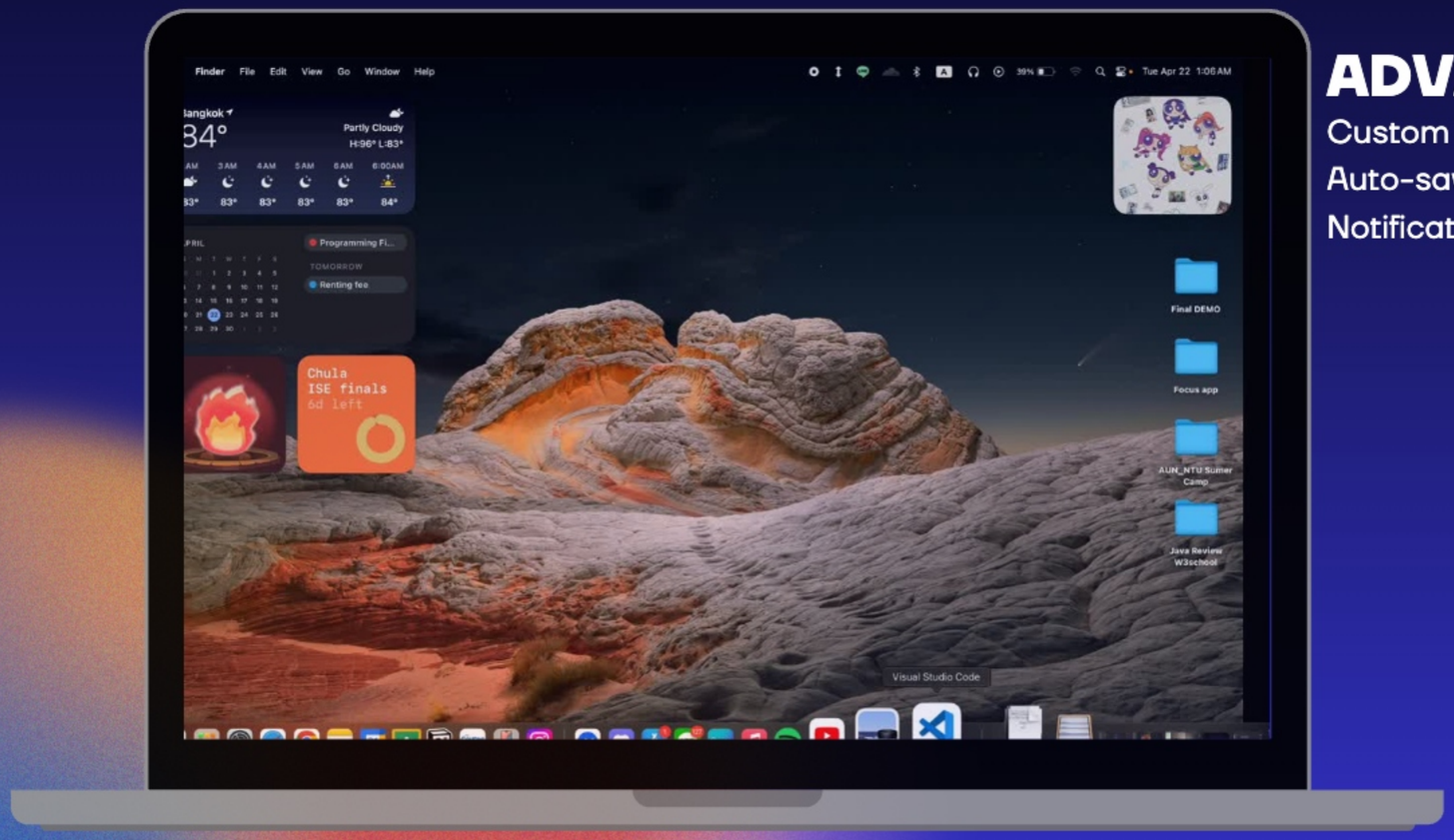
LIVE

DEMOSTRATION

TEAM MEMBER
SAI, CAPPI & KHOI

Tech Stack
Java, Java Swing

FOCUS TRACK
PRODUCTIVITY APP



ADVANCED Features

Custom Timer Strategy

Auto-save or File persistence

Notification Volume Control

LIVE
DEMOSTRATION

TEAM MEMBER
SAI, CAPPI & KHOI

Tech Stack
Java, Java Swing

FOCUS TRACK

STAYED LOCKED IN

TEAM MEMBER
SAI, CAPPI & KHOI

Tech Stack
Java, Java Swing