Project Title: ShadeSync

Team Name: The Lightbenders

Team Members:

- Matthew Dauria – Collaborative Development Role
- Jake Sussner – Collaborative Development Role
- Brian Quintero – Collaborative Development Role

(Our team works together across hardware, software, and testing. Each member brings unique strengths but all roles overlap.)

GitHub Repository URL: https://github.com/brianquintero13/Senior-Capstone

Trello Board URL: https://trello.com/b/8fFUBatY/shadesync

Diagrams

Create the required diagrams to help you visualize and communicate the design of your system. Each diagram should be in .png or .jpg format, properly named (e.g., context.png, dataflow.png), and included in a ZIP file for submission.

1. **Context diagram**
   Create a Context Diagram that shows the entire system and its interactions with external entities (users,
   is clear and shows all important system interactions.

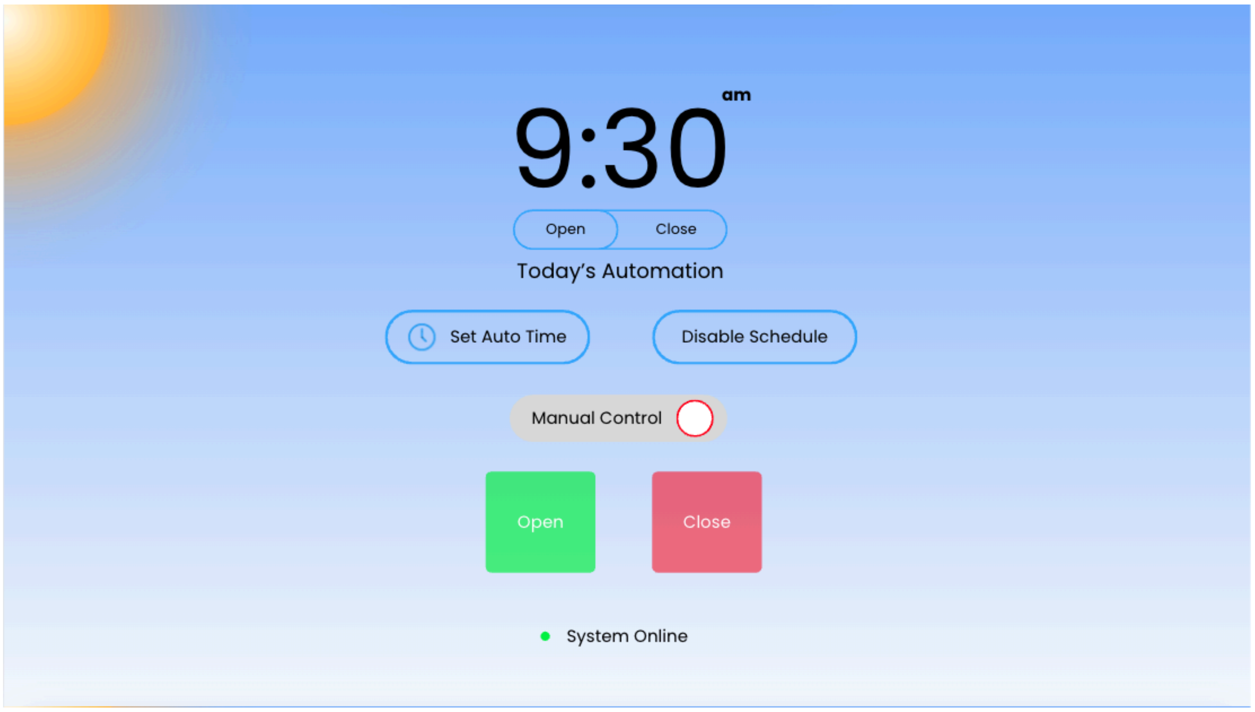
2. **Data flow diagram**
   Create a Data Flow Diagram (DFD) showing the flow of data within your system. Choose one of the data
   and expand it to show how the input becomes output. The diagram should have sufficient detail and clarity.

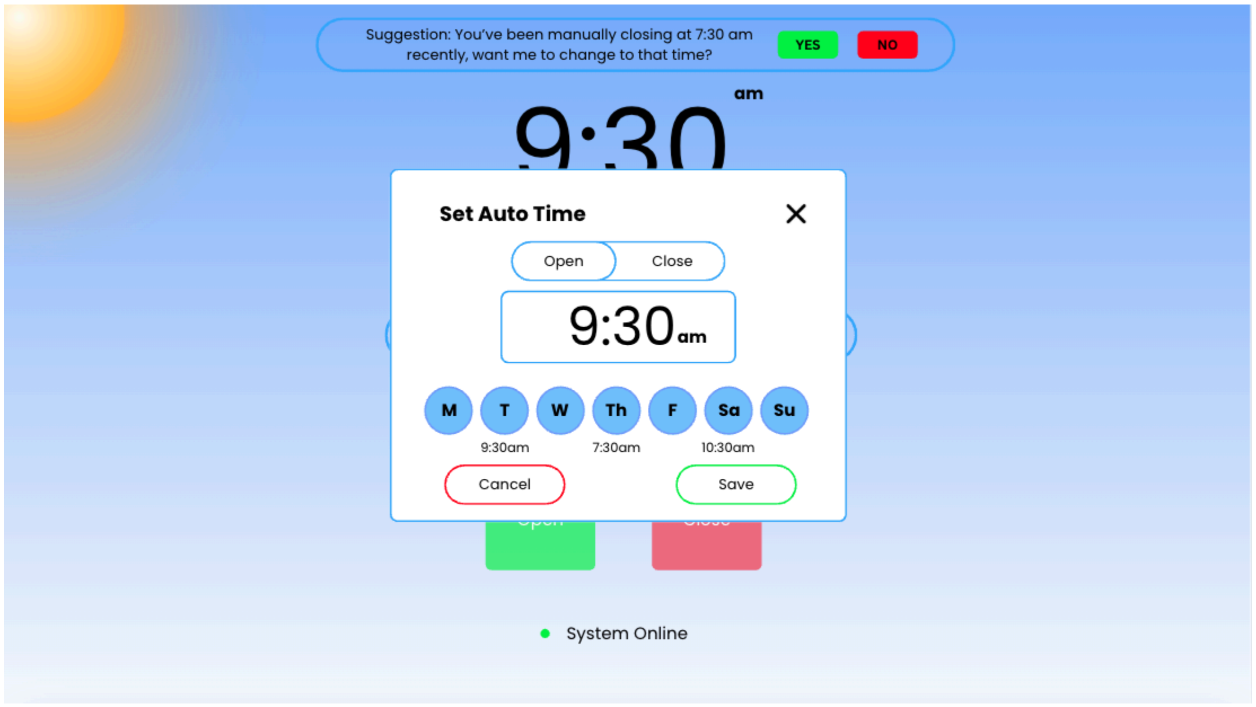
3. **Use case diagram**
   Create a Use Case Diagram for one of your system's features.
   Include at least five use cases that represent different user interactions with the system. Clearly define the actors and the use cases that describe how users interact with your system.

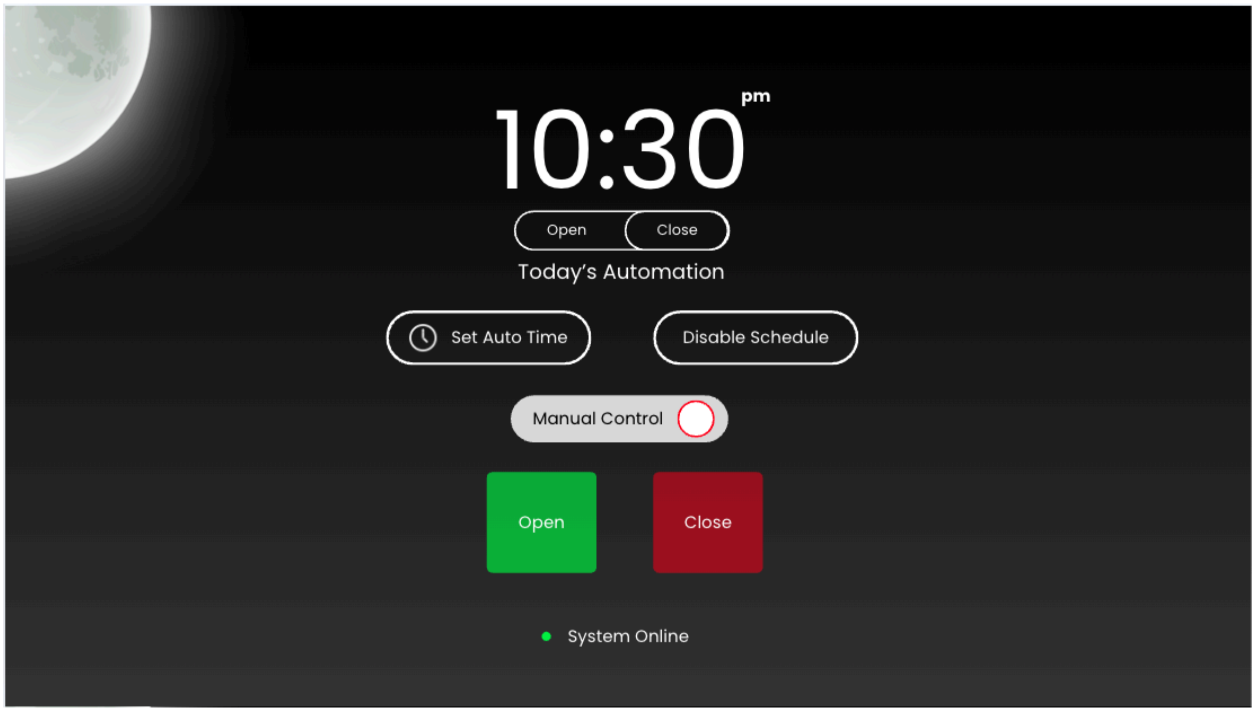
4. **Sequence diagram**
   Create a Sequence Diagram for one of the use cases from your Use Case Diagram. This diagram should show the full interactions required for the selected use case, detailing the messages exchanged between objects

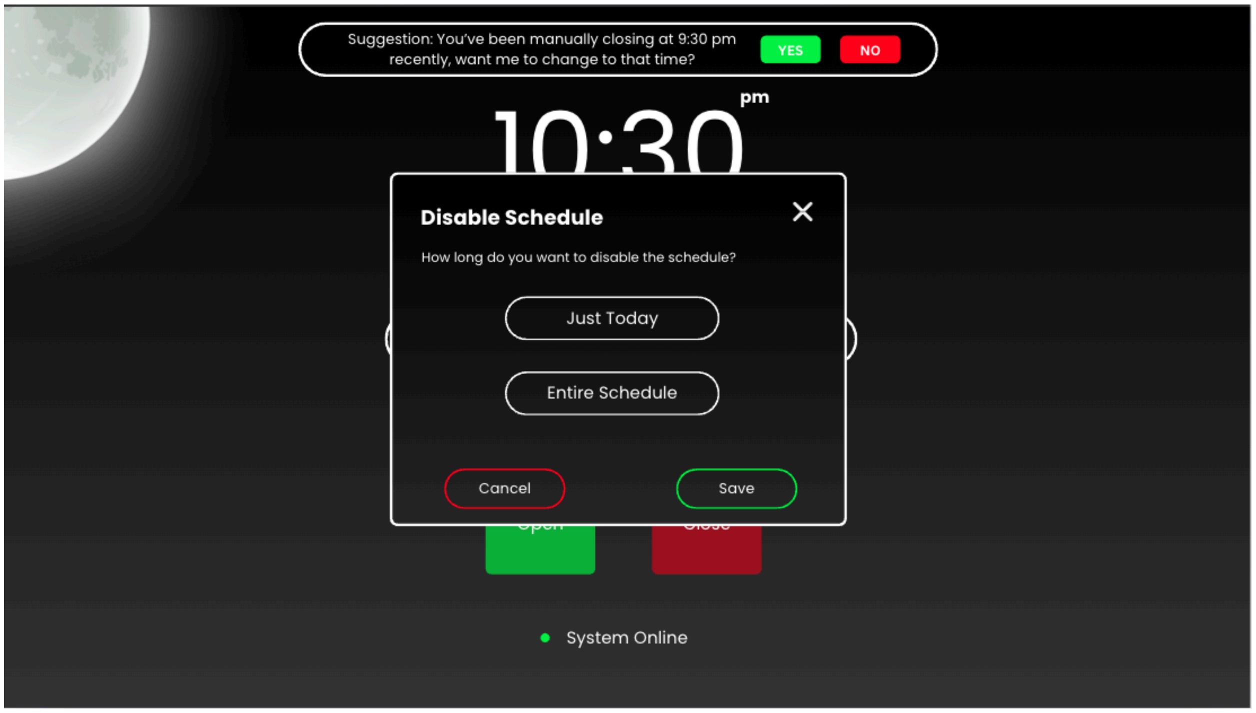
5. **Class diagram**
   Create a Class Diagram that shows the classes and methods used in your Sequence Diagram, as well a
   your system needs. Ensure there are at least four user-defined types represented, showing the structure of your system's objects.

UI mock-ups



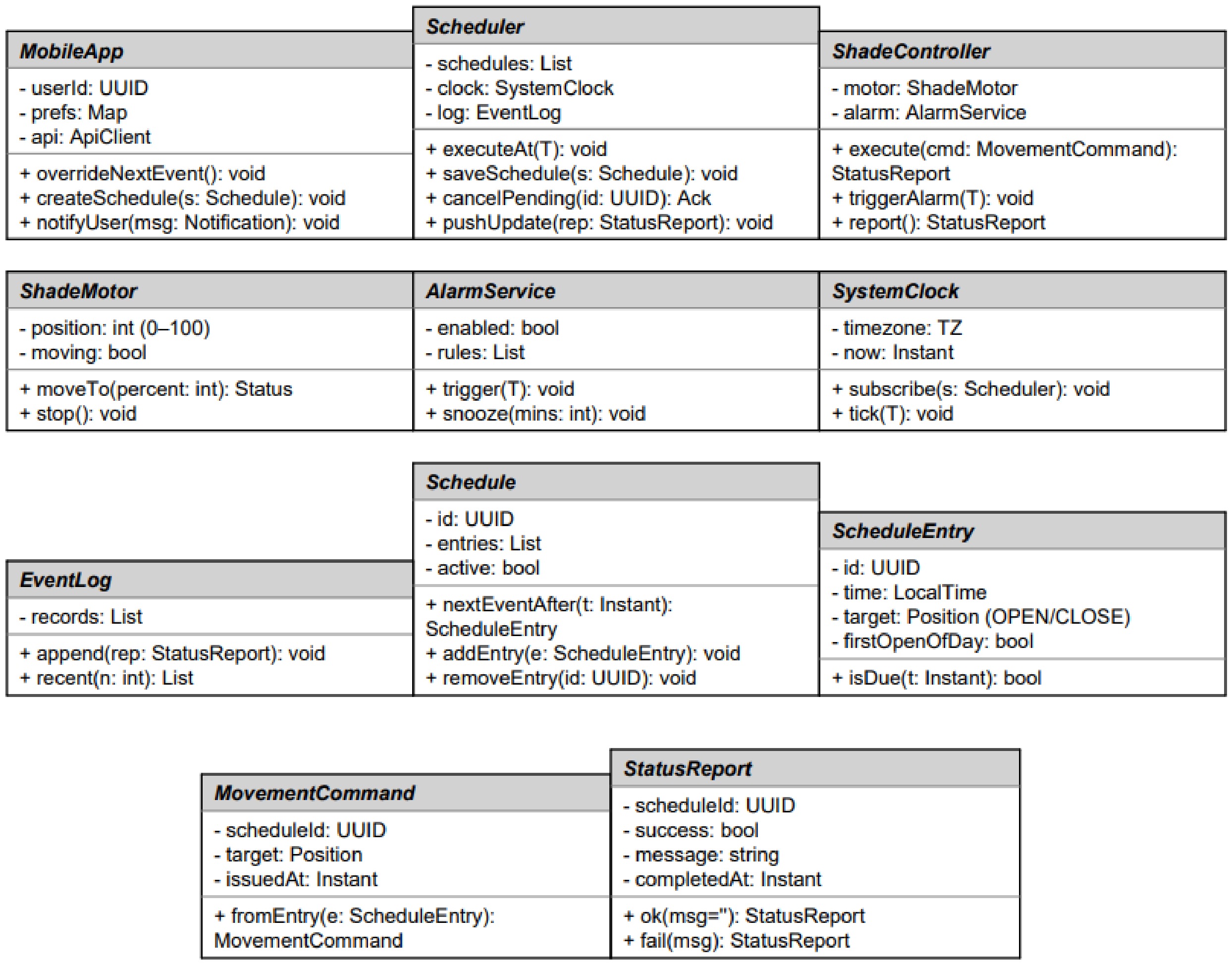

Suggestion: You've been manually closing at 7:30 am recently, want me to change to that time?


10:30 pm
Open
Close
Today's Automation
Set Auto Time
Disable Schedule
Manual Control
Open
Close
System Online


Suggestion: You've been manually closing at 9:30 pm recently, want me to change to that time?
YES
NO
10:30 pm
Disable Schedule
How long do you want to disable the schedule?
Just Today
Entire Schedule
Cancel
Save
System Online

Class Diagram

ShadeSync Class Diagram



MobileApp
- userId: UUID
- prefs: Map
- api: ApiClient

+ overrideNextEvent(): void
+ createSchedule(s: Schedule): void
+ notifyUser(msg: Notification): void

Scheduler
- schedules: List
- clock: SystemClock
- log: EventLog

+ executeAt(T): void
+ saveSchedule(s: Schedule): void
+ cancelPending(id: UUID): Ack
+ pushUpdate(rep: StatusReport): void

ShadeController
- motor: ShadeMotor
- alarm: AlarmService

+ execute(cmd: MovementCommand): StatusReport
+ triggerAlarm(T): void
+ report(): StatusReport

ShadeMotor
- position: int (0–100)
- moving: bool

+ moveTo(percent: int): Status
+ stop(): void

AlarmService
- enabled: bool
- rules: List

+ trigger(T): void
+ snooze(mins: int): void

SystemClock
- timezone: TZ
- now: Instant

+ subscribe(s: Scheduler): void
+ tick(T): void

EventLog
- records: List

+ append(rep: StatusReport): void
+ recent(n: int): List

Schedule
- id: UUID
- entries: List
- active: bool

+ nextEventAfter(t: Instant): ScheduleEntry
+ addEntry(e: ScheduleEntry): void
+ removeEntry(id: UUID): void

ScheduleEntry
- id: UUID
- time: LocalTime
- target: Position (OPEN/CLOSE)
- firstOpenOfDay: bool

+ isDue(t: Instant): bool

MovementCommand
- scheduleId: UUID
- target: Position
- issuedAt: Instant

+ fromEntry(e: ScheduleEntry): MovementCommand

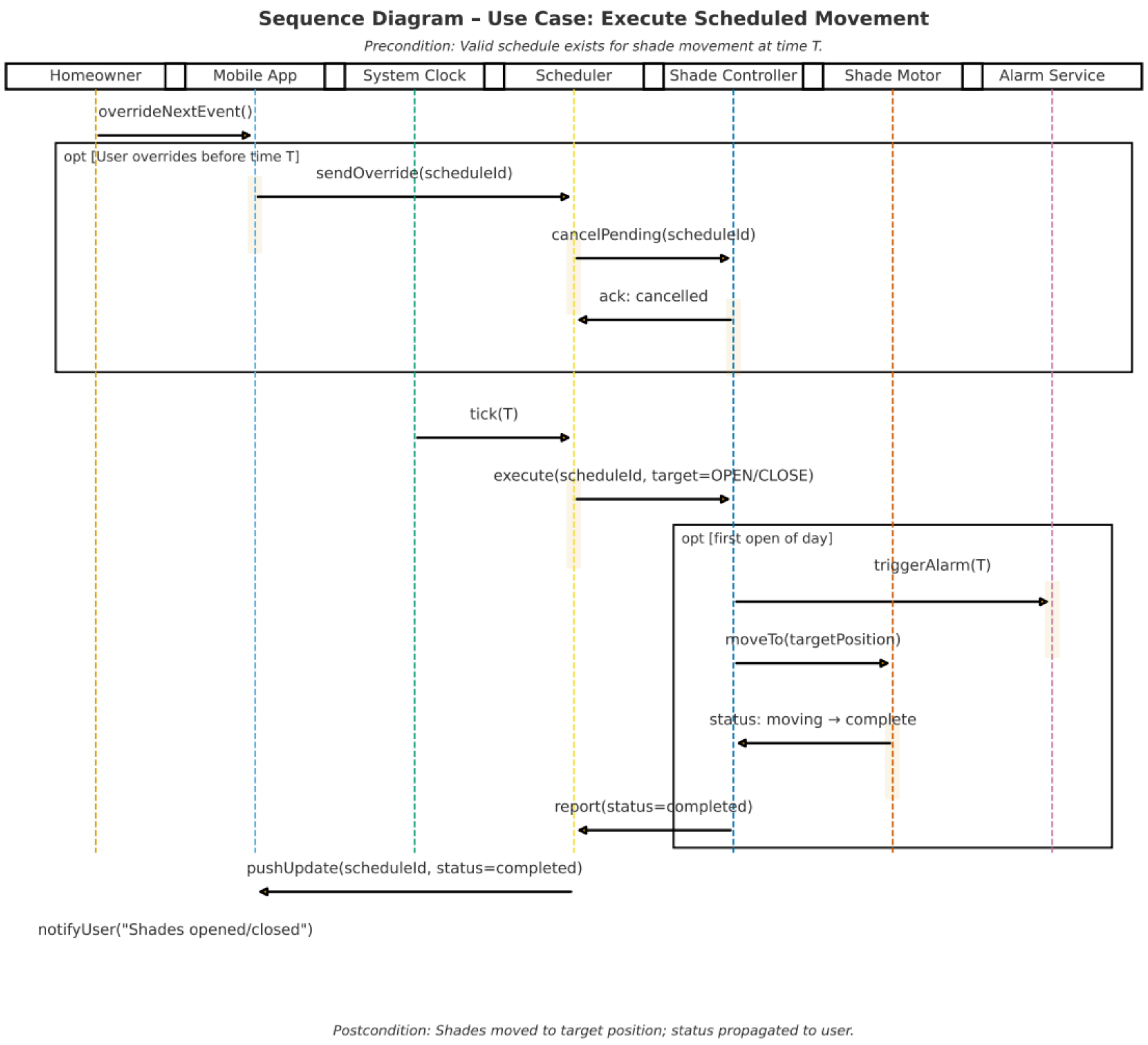
StatusReport
- scheduleId: UUID
- success: bool
- message: string
- completedAt: Instant

+ ok(msg=''): StatusReport
+ fail(msg): StatusReport

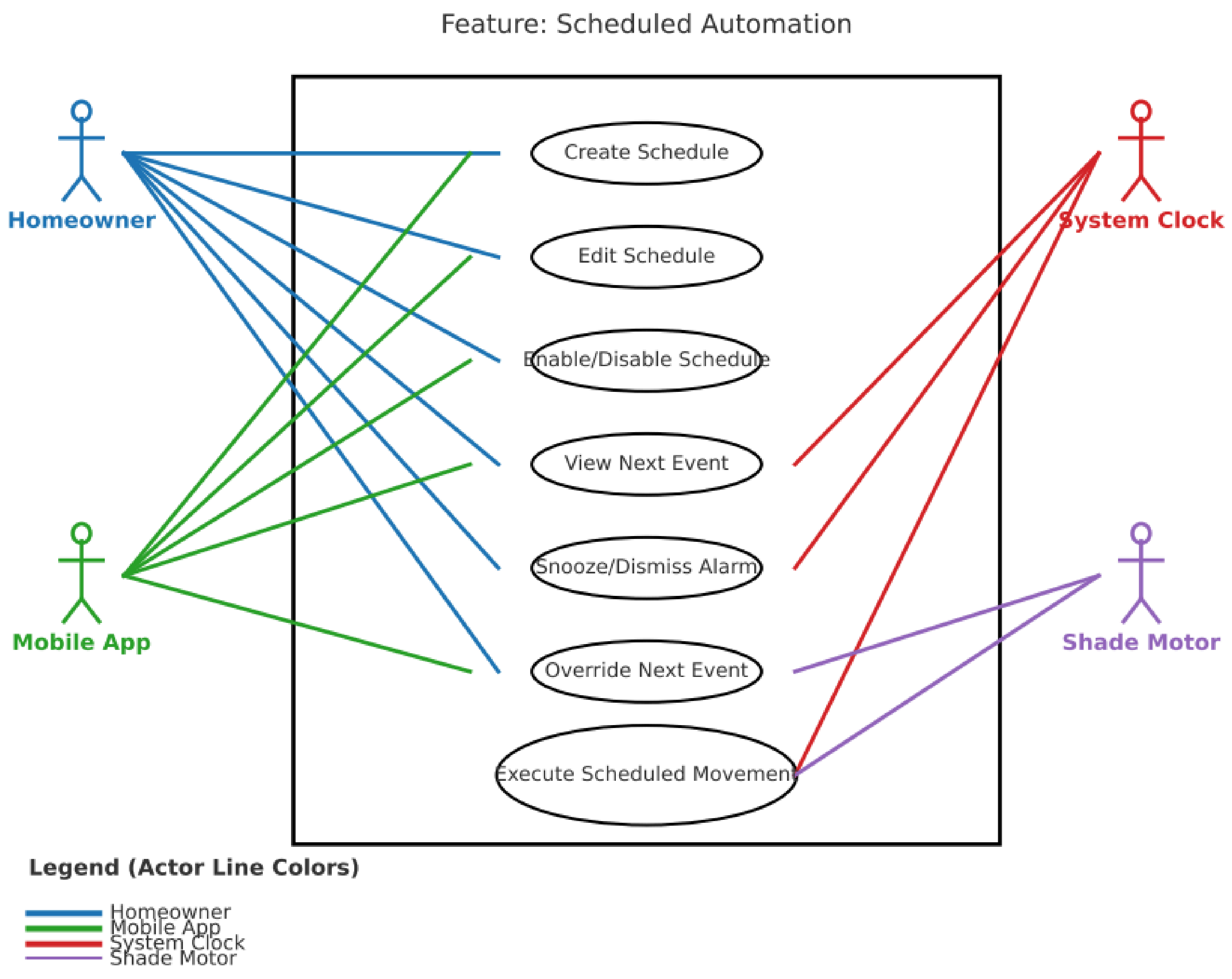
Relationships (conceptual):

MobileApp → Scheduler (creates/overrides)
Scheduler → ShadeController (executes)
ShadeController → ShadeMotor (commands, uses)
ShadeController → AlarmService (uses)
Scheduler → EventLog (writes/reads)
Scheduler → SystemClock (subscribes)
Scheduler → Schedule (reads/writes)
Schedule → ScheduleEntry (uses)
ScheduleEntry → MovementCommand (builds command)
MovementCommand → ShadeController (returns report)

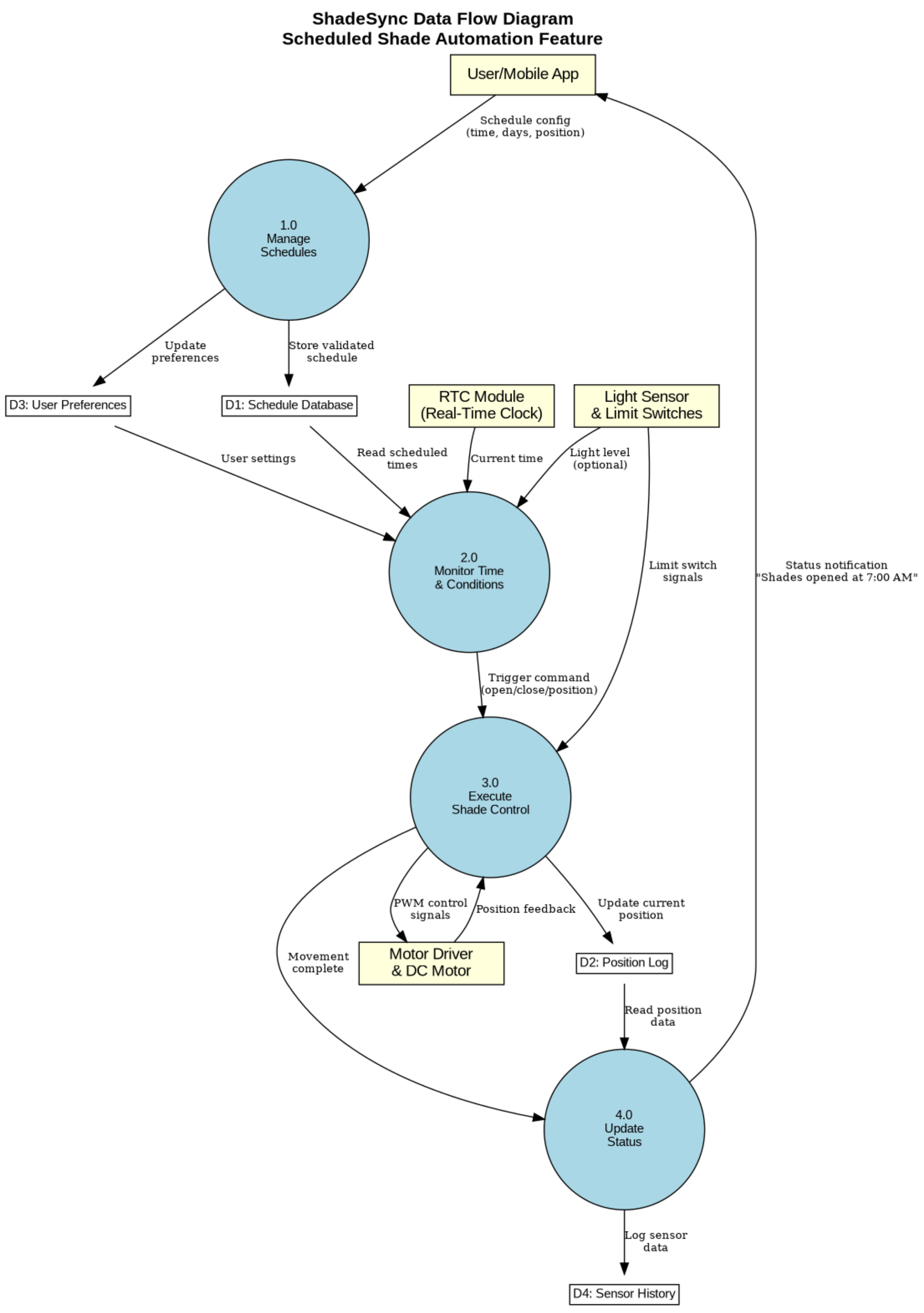
Sequence Diagram


Sequence Diagram – Use Case: Execute Scheduled Movement
Precondition: Valid schedule exists for shade movement at time T.
Homeowner
Mobile App
System Clock
Scheduler
Shade Controller
Shade Motor
Alarm Service
overrideNextEvent()
opt [User overrides before time T]
sendOverride(scheduleId)
cancelPending(scheduleId)
ack: cancelled
tick(T)
execute(scheduleId, target=OPEN/CLOSE)
opt [first open of day]
triggerAlarm(T)
moveTo(targetPosition)
status: moving → complete
report(status=completed)
pushUpdate(scheduleId, status=completed)
notifyUser("Shades opened/closed")
Postcondition: Shades moved to target position; status propagated to user.

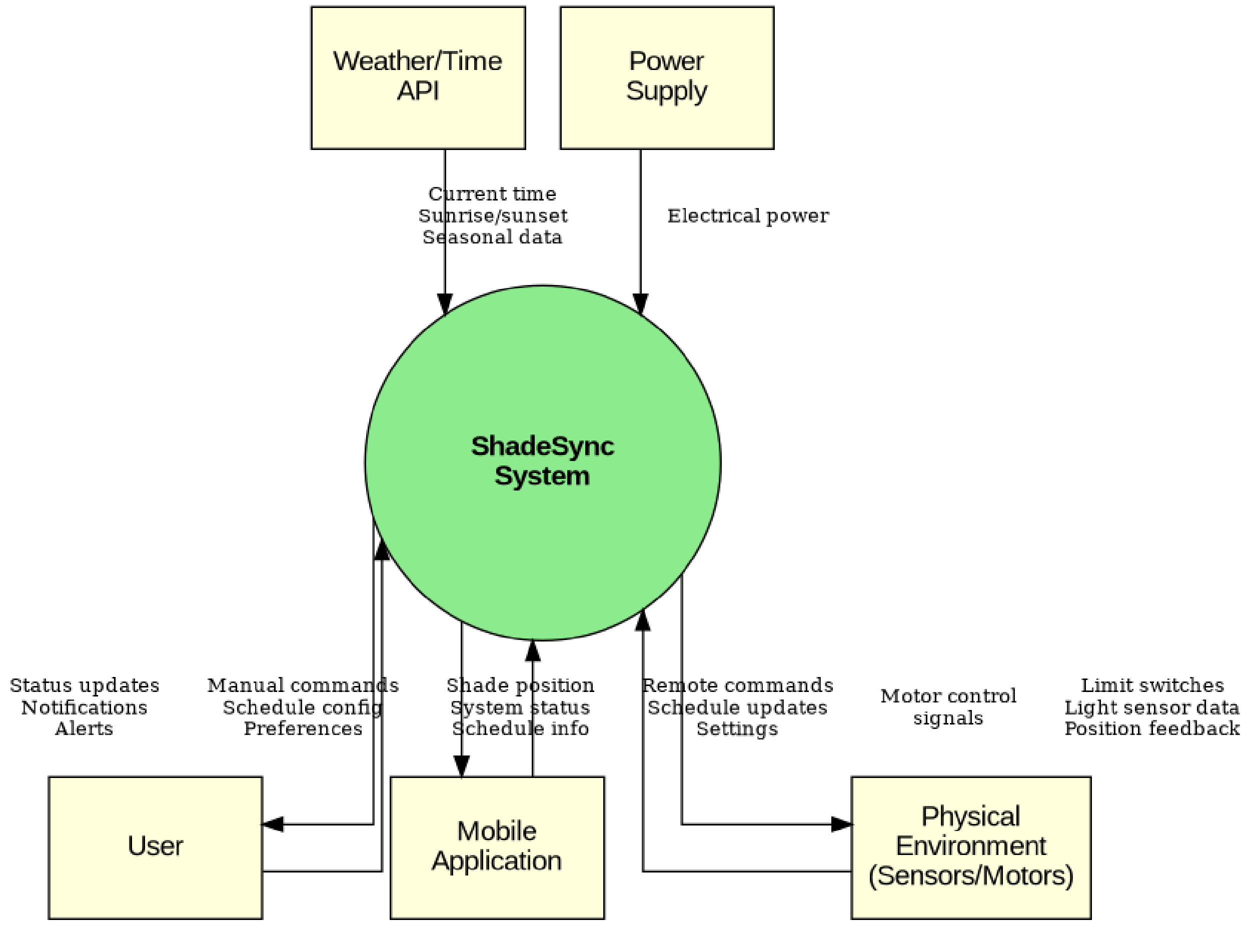
Use Case Diagram



Data Flow Diagram



Context Diagram


Weather/Time API
Power Supply
Current time
Sunrise/sunset
Seasonal data
Electrical power
ShadeSync System
Status updates
Notifications
Alerts
Manual commands
Schedule config
Preferences
Shade position
System status
Schedule info
Remote commands
Schedule updates
Settings
Motor control signals
Limit switches
Light sensor data
Position feedback
User
Mobile Application
Physical Environment (Sensors/Motors)