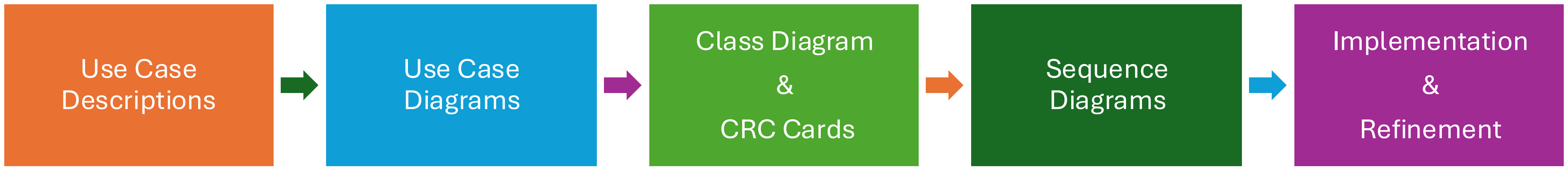
SafeHome Control Hub

Sydnie Pittman, Darren Harvey, Arianna Banton, Hassan Stewart

Objective

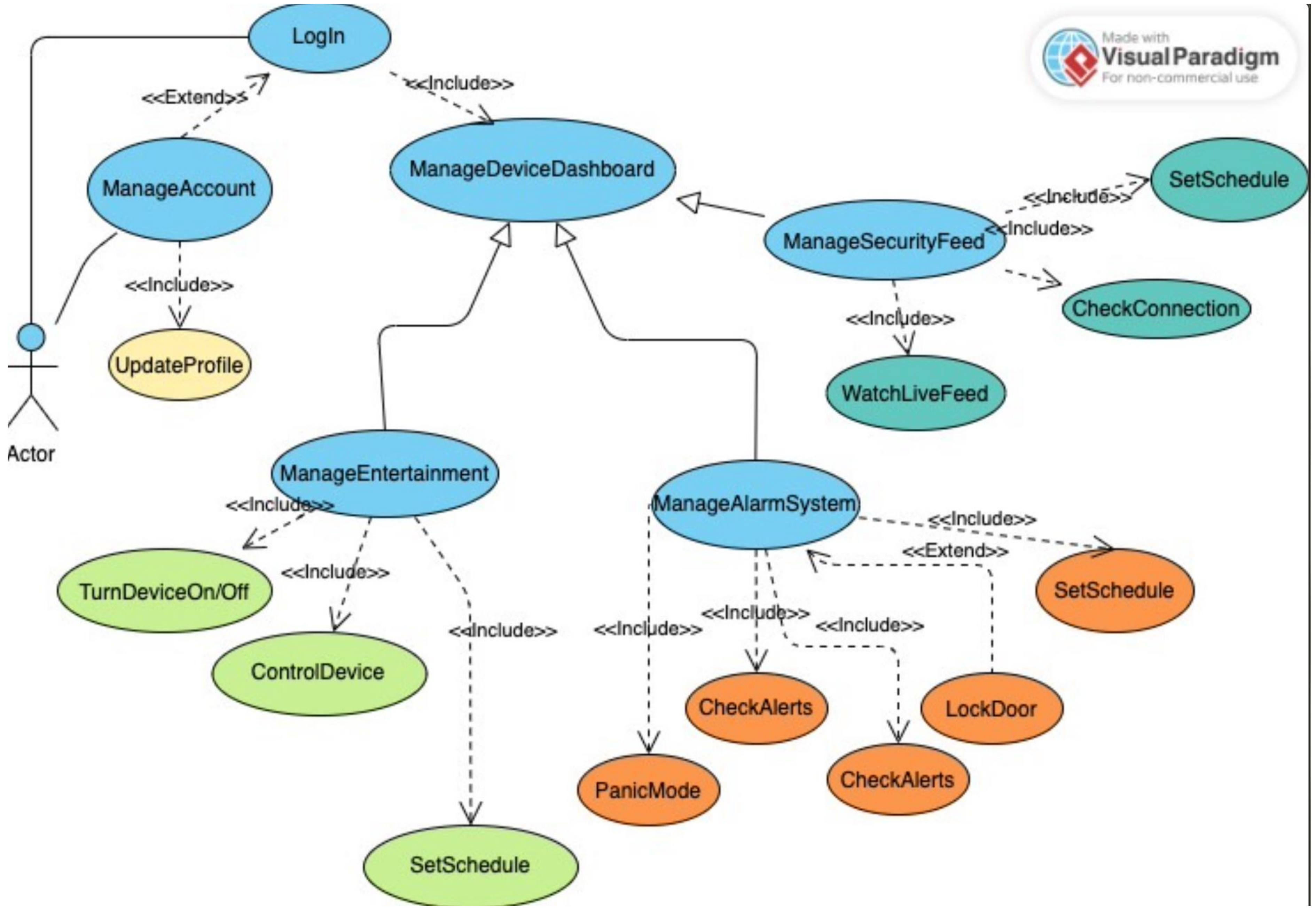
- Design and simulate a smart home control system through centralized control of IoT devices
- Use object-oriented modeling techniques
- Our system attempts to allow users to:
  - Control and monitor devices remotely
  - Automate routines
  - Grant guest access
  - Respond to emergencies quickly

Our Process


Use Case Descriptions
Use Case Diagrams
Class Diagram & CRC Cards
Sequence Diagrams
Implementation & Refinement

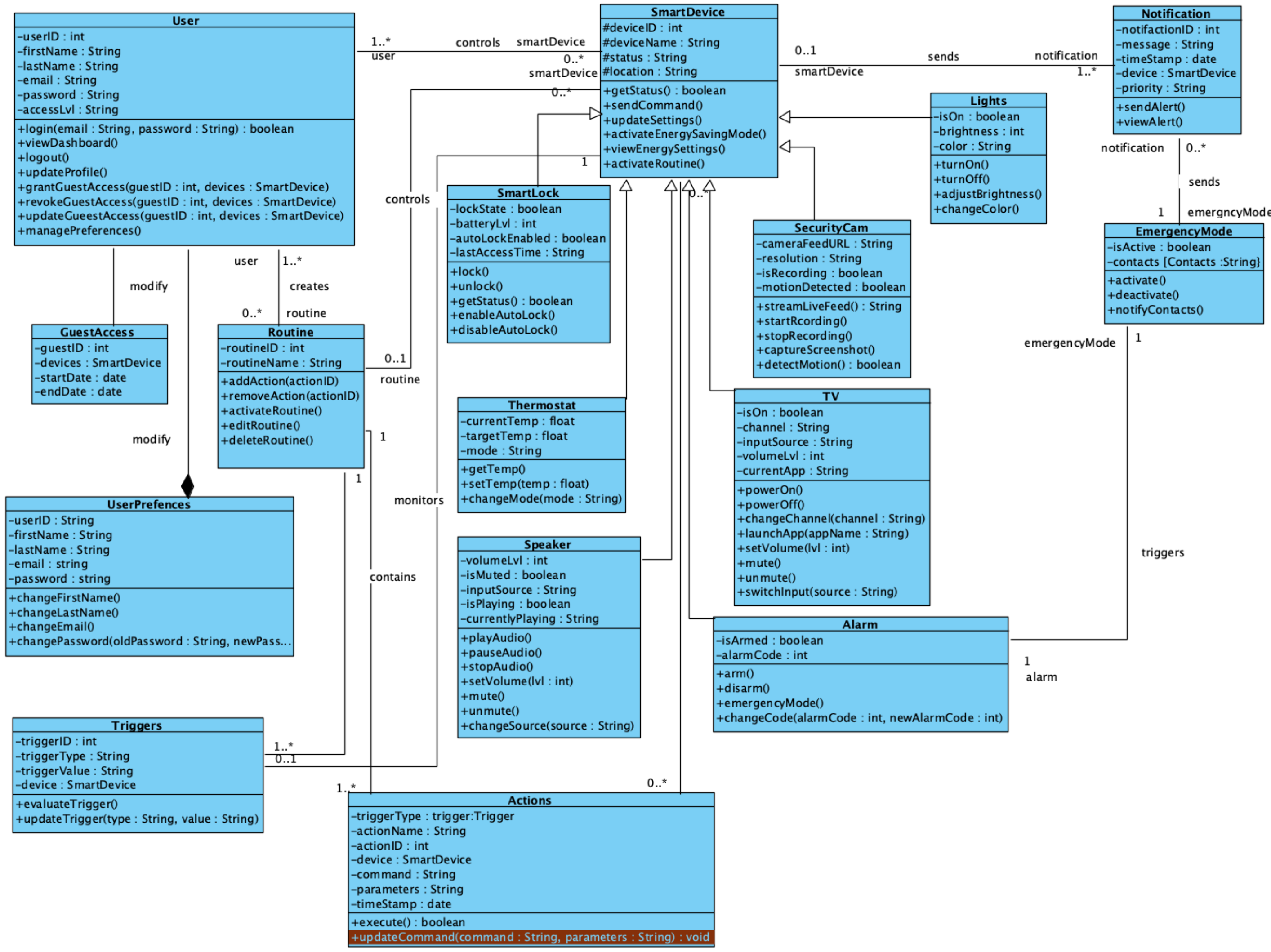
Key Use Cases

- Manage User Preferences
  - Allow user profile personalization and security
  - Update name, email, password, etc.
- Manage Smart Devices (TV, Lights, Locks, etc.)
  - Turn devices on, off, manage routines
- Manage Guest Access
  - Grant/revoke guest access to devices
- Create/manage device routines


LogIn
<<Extend>>
<<Include>>
ManageAccount
ManageDeviceDashboard
<<Include>>
UpdateProfile
Actor
ManageSecurityFeed
<<Include>>
<<Include>>
SetSchedule
CheckConnection
<<Include>>
WatchLiveFeed
ManageEntertainment
<<Include>>
TurnDeviceOn/Off
<<Include>>
ControlDevice
<<Include>>
SetSchedule
ManageAlarmSystem
<<Include>>
<<Extend>>
SetSchedule
<<Include>>
<<Include>>
<<Include>>
PanicMode
CheckAlerts
CheckAlerts
LockDoor
Made with Visual Paradigm For non-commercial use

Class Diagram

- User Class
    - Central actor of system
    - Should be able to manage their account and devices
- SmartDevice Class
    - Abstract class
    - Multiple "is-a" inherited subclasses for scalability (Speaker, TV, Lights, etc.
- Routine Class
    - Manage routine actions
    - Manage routine triggers
- Emergency/Notification Classes



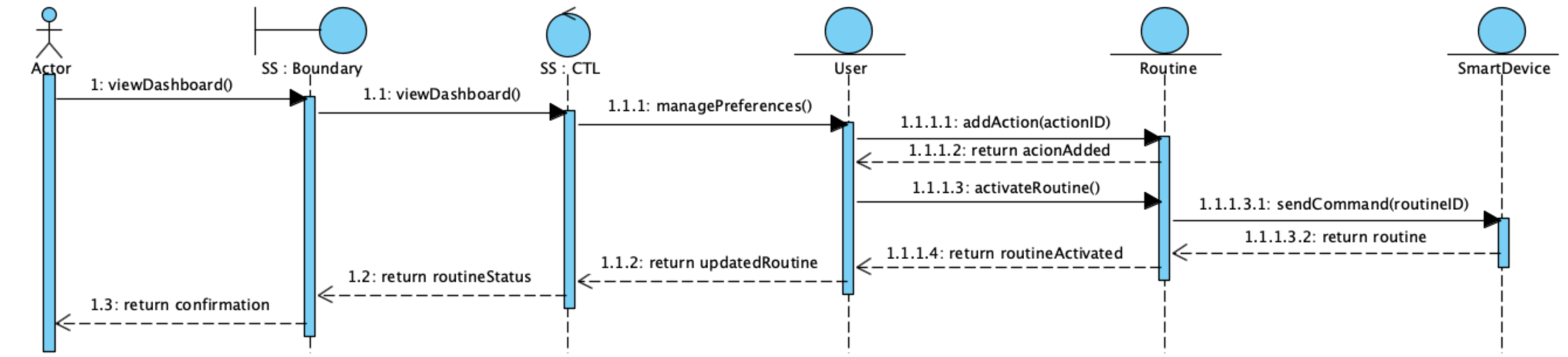
Sequence Diagrams

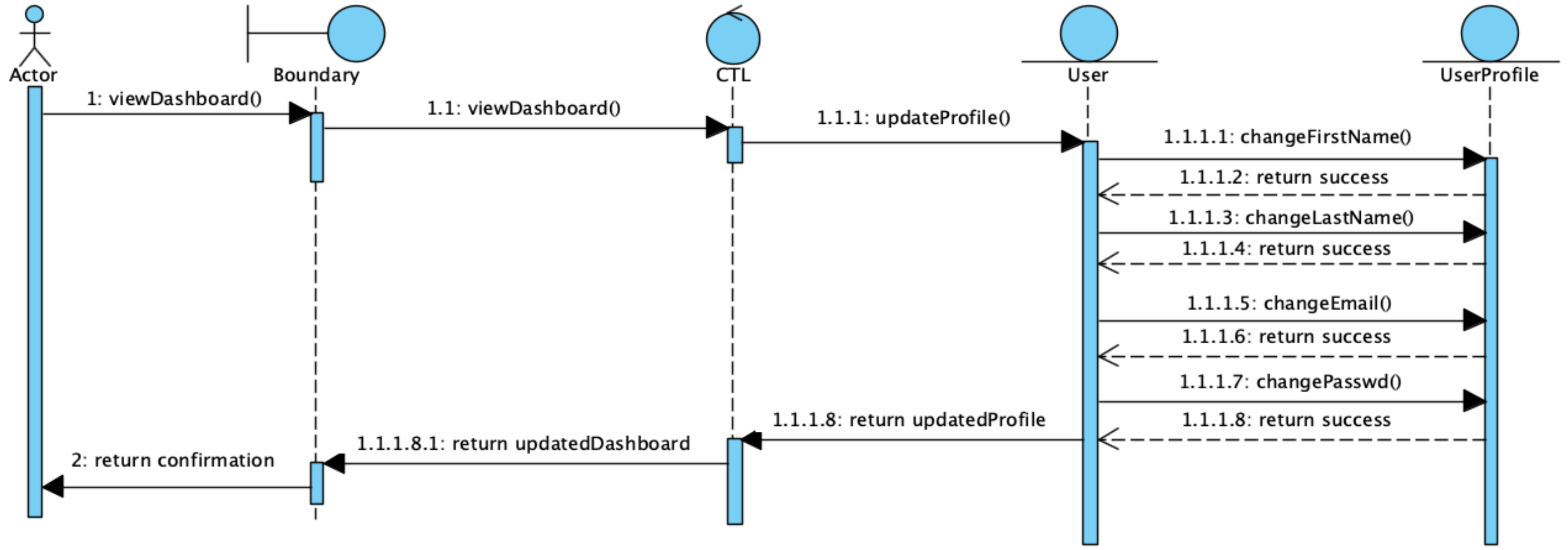
- Update Profile
  Actors:
  - Boundary → Control (CTL) → User → UserProfile

  Flow:
  - viewDashboard() from Boundary to Control
  - updateProfile() from Control to User
  - User calls methods to:
    - changeFirstName()
    - changeLastName()
    - changeEmail()
    - changePassword()
  - Each method returns success.
  - Dashboard gets updated.

- Set Schedule
  Actors:
  - Boundary → Control (CTL) → User → Routine

  Flow:
  - viewDashboard() from Boundary to Control
  - managePreferences() from Control to User
  - User does:
    - addAction(actionID)
    - activateRoutine()
  - Routine processes sendAction()
  - Responses are sent back step-by-step (success, updatedRoutine, confirmation)


Actor
SS : Boundary
SS : CTL
User
Routine
SmartDevice
1: viewDashboard()
1.1: viewDashboard()
1.1.1: managePreferences()
1.1.1.1: addAction(actionID)
1.1.1.2: return acionAdded
1.1.1.3: activateRoutine()
1.1.1.3.1: sendCommand(routineID)
1.1.1.3.2: return routine
1.1.1.4: return routineActivated
1.1.2: return updatedRoutine
1.2: return routineStatus
1.3: return confirmation



Entity Classes

Device Entities

- All extend SmartDevice and add custom behavior
- TV, Lights, Speaker, SmartLock, Thermostat

User Entity

- Represents a registered smart home user

Routine/Actions

- Routine: represents series of scheduled/triggered actions
- Actions: Represents specific task executed on a device

GuestAccess/UserPreferences

- Temporarily grant access to devices
- Store customizable profile settings

Control Classes

EmergencyController

- Activates and deactivates emergency mode
- Notifies emergency contacts
- Unlocks doors, turns on alarms/cameras during emergencies

DeviceController

- Adds and removes devices
- Sends commands (e.g., "lock", "on", "off")
- Displays device statuses
- Activates energy saving mode across all devices

GuestController

- Grants, updates, and revokes guest access to devices
- Manages temporary guest permissions

RoutineController

- Creates, edits, and deletes routines
- Adds or removes actions from routines
- Activates routines based on triggers

Boundary Class

- Acts as the user interface for the smart home application
- Displays menus and reads user inputs via the console
- Delegates commands to respective controller classes
- Provides a structured flow for user interaction

Lessons Learned

Planning is Everything

- Starting with a clear narrative and use cases aligned our team and helped avoid confusion later

Design First, Then Build

- Creating class diagrams first helped us identify reusable components like SmartDevice

Relationships Matter

- Understanding associations (e.g., User ↔ SmartDevice) made our system more realistic and scalable

A Step Further...

- File-based storage of routines and devices
- Scheduled triggers and time-based automation
- GUI or web-based interface
- Enhanced notification system

Source Code

https://github.com/sydniepittman/CSCI3320_ProjectSHCH