PROJECT GREENTHUMB:
SEMI-AUTONOMOUS IOT GARDENING SYSTEM

Grand Valley State University
College of Computing

Khiem Hoang Nguyen

Advisor: Erik Fredericks

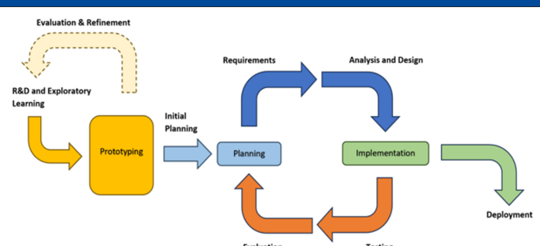
Introduction

- Project Greenthumb was a semi-automated IoT gardening system that is comprised three subsystems:
  - 1) **Embedded system:** A programmed Arduino with sensors and devices that read sensors, published data to the broker, and responded to sensor threshold parameters
  - 2) **Frontend web application:** The UI of the application used to display sensor data and chat with agents about the project
  - 3) **Backend REST API**: Abstracted the logic behind the agentic chat messenger and sensor data processes and provided endpoints to those services
- The semi-automated IoT gardening system had the capability to grow plants with short growing periods (2 weeks), such as cat grass, from seed to maturity without any human intervention
- Project Greenthumb begun a few months before the Fall 2025 semester. Exploratory learning & prototyping for the embedded system started in July 2025 and agentic design & orchestration in Aug. 2025. Iterative developmental lifecycle begun Sept. 2025

Developmental Toolbox

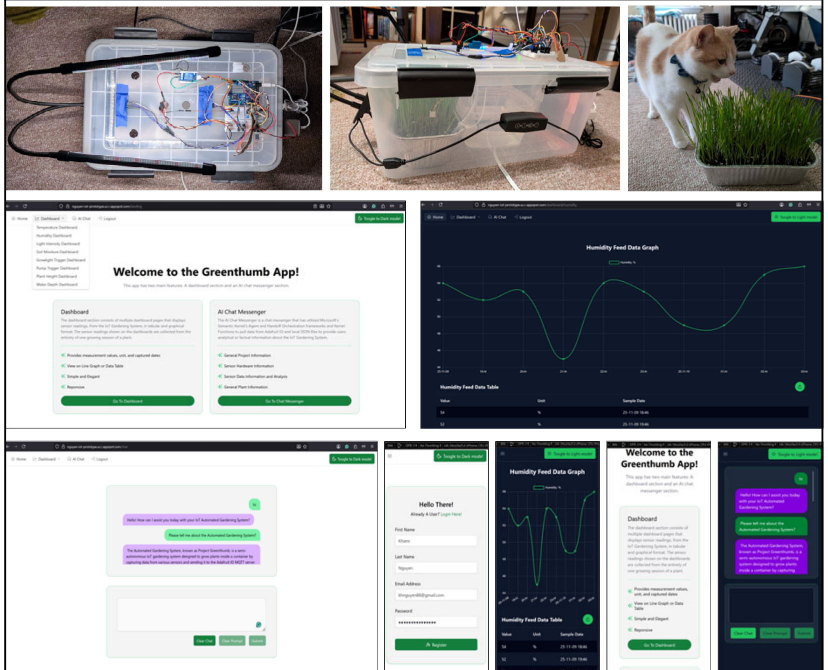
Visual Studio	MS Kernel Memory SDK
Visual Studio Code	C#
Arduino IDE	Typescript
Angular Framework	C++
PrimeNG Component Library	Adafruit IO
PrimeFlex CSS Utility Library	Microsoft Azure
PrimeBlock UI Library	Googe Firebase
MS Semantic Kernel SDK	Google Cloud Platform

Developmental Lifecycle



The project followed a hybrid software developmental process, which combined the Exploratory and Iterative Developmental models. The initial part of the process relied heavily on prototyping and the development of thruway applications to build the knowledge on concepts and topics that were never covered in the graduate program's academic curriculum or used at work, such as embedded design and circuitry, IoT, and agentic design. The exploratory process helped refine the project's scope and feasibility based on the knowledge gained and contributed to the initial plans for the project's iterative developmental lifecycle.

Results/Implementation



Try the App Here

Major Challenges and Solutions

- **Embedded Hardware & Application :**
  - Had to limited knowledge in circuitry, the C++ language, and IoT but learned and applied them to the application in a short timeframe, through reading online forums, documentations, and prototyping
  - Encountered sensors that provided faulty readings, required the development of a checklist of every root causes encountered and its utilization to streamline debugging process
  - VS Code IDE could not find Arduino Library, and developed workaround by coding in VS Code IDE and compiling and debugging code in Arduino IDE.
- **Frontend Application:**
  - Angular UI Component Library, PrimeNG, updated it's theming system and styles customization process, and had poor documentations, but learned new theming system through online resources.
- **Backend Application:**
  - Had to limited knowledge in agentic design and orchestration, but learned and applied them into application through reading Microsoft Semantic Kernel documentation and building prototypes.
  - Cloud Run deployment of the backend application onto the cloud required additional process of generating a Docker image of the application and uploading it onto GCP's Artifacts Registry

Technical Details

- **Embedded Hardware & Application :**
  - Embedded system comprised of a programmed Arduino, multiple sensors (DHT11, Ultrasonic Sonar Sensors, Photoresistors, and Soil Conductivity Sensor), 5v water pump, 5v growlight, and relay devices.
  - Adafruit MQTT Library was used to publish sensor data to Adafruit IO
  - Concurrent timers were used to trigger staggered events such as sampling sensors, publishing data, and activating / deactivating devices
- **Frontend Application:**
  - Had four main pages: Landing, Dashboard, Chat Messenger, and Auth
  - Developed using Angular and Prime libraries
  - UI was responsive & worked with various screens; Had light/dark mode
  - Utilized Google Firebase Authentication for user login, logout, & signup
- **Backend Application:**
  - Provided a service that was used to retrieve data from Adafruit IO and clean it up for external (frontend) and internal use.
  - Provided a service that used to take user prompts & pass them through an agentic handoff orchestration, utilizing a remote LLM, to best provide an appropriate response based on the topic of the prompts.
    - The agents and handoff orchestration were created using Microsoft's Semantic Kernel SDK
- **Cloud Services:**
  - Google Cloud Platform deployed the frontend & backend applications
  - Google Firebase provided authentication for frontend application
  - Azure hosted and deployed LLM Model used in backend application
  - Adafruit IO was used as MQTT broker and a database service

Proposed Future Improvements

- **Embedded Hardware & Application :**
  - Upgrade the ultrasonic sonar sensors with water meter sensor, LiDAR sensor, and camera to improve accuracy measuring plant growth and water supply levels.
  - Refine existing code, continue support for the ultrasonic sensor, and implement code for new sensors and camera
  - Adjust concurrent timers and frequency of sensor reading and data publication to Adafruit IO (MQTT Broker)
  - Enable MQTT subscription features to allow users to adjust parameters while the Gardening System is running
- **Frontend Application:**
  - Add Embedded System Settings page for embedded application parameters customization
  - Add a Users Settings page for web application customization
- **Backend Application:**
  - Revise and make the Chat Completion Service extensible by giving users ability to generate and configure agents and orchestration through YAML template and config files; including LLM model selection
  - Create an extensible Data Service and interface to support different database services and connection, including local database, that can be selected via a configuration file.
  - Implement new service to accept and process images from camera