Senior Center App

Group 5 - Rial Johnson, Alexandr Matveyev, Nick Hager, Brandon May, Rusty Clayton

Project Overview

The Belgrade Senior Center has approximately 200 members. Our goal was to create a web application to help automate their processes through the following features:

Project Overview

The Belgrade Senior Center has approximately 200 members. Our goal was to create a web application to help automate their processes through the following features:

1) Capture and store member and volunteer information (name, address, membership status, etc.).

Project Overview

The Belgrade Senior Center has approximately 200 members. Our goal was to create a web application to help automate their processes through the following features:

1) Capture and store member and volunteer information (name, address, membership status, etc.).
2) Create and manage activities/events, including member enrollment/attendance.

Project Overview

The Belgrade Senior Center has approximately 200 members. Our goal was to create a web application to help automate their processes through the following features:

1) Capture and store member and volunteer information (name, address, membership status, etc.).
2) Create and manage activities/events, including member enrollment/attendance.
3) Export relevant data to 3rd-party programs (i.e. Excel) for accurate grant writing and reporting.

Technologies

To build this project, we used the following technologies:

Technologies

To build this project, we used the following technologies:

1) Angular - TypeScript-based open-source web application framework, developed by the Angular Team at Google.

Technologies

To build this project, we used the following technologies:

1) Angular - TypeScript-based open-source web application framework, developed by the Angular Team at Google.
2) Amazon Web Services (AWS) - on-demand cloud computing platforms.

Technologies

To build this project, we used the following technologies:

1) Angular - TypeScript-based open-source web application framework, developed by the Angular Team at Google.
2) Amazon Web Services (AWS) - on-demand cloud computing platforms.
    a) Amazon S3 (simple storage service) - provides object storage through a web service (AWS) interface.

Technologies

To build this project, we used the following technologies:

1) Angular - TypeScript-based open-source web application framework, developed by the Angular Team at Google.
2) Amazon Web Services (AWS) - on-demand cloud computing platforms.
    a) Amazon S3 (simple storage service) - provides object storage through a web service (AWS) interface.
    b) Amazon DynamoDB (database) - NoSQL database service that supports key-value and document data structures offered by AWS.

Technologies

To build this project, we used the following technologies:

1) Angular - TypeScript-based open-source web application framework, developed by the Angular Team at Google.
2) Amazon Web Services (AWS) - on-demand cloud computing platforms.
    a) Amazon S3 (simple storage service) - provides object storage through a web service (AWS) interface.
    b) Amazon DynamoDB (database) - NoSQL database service that supports key-value and document data structures offered by AWS.
3) Travis CI - Continuous integration service used to build and test software.

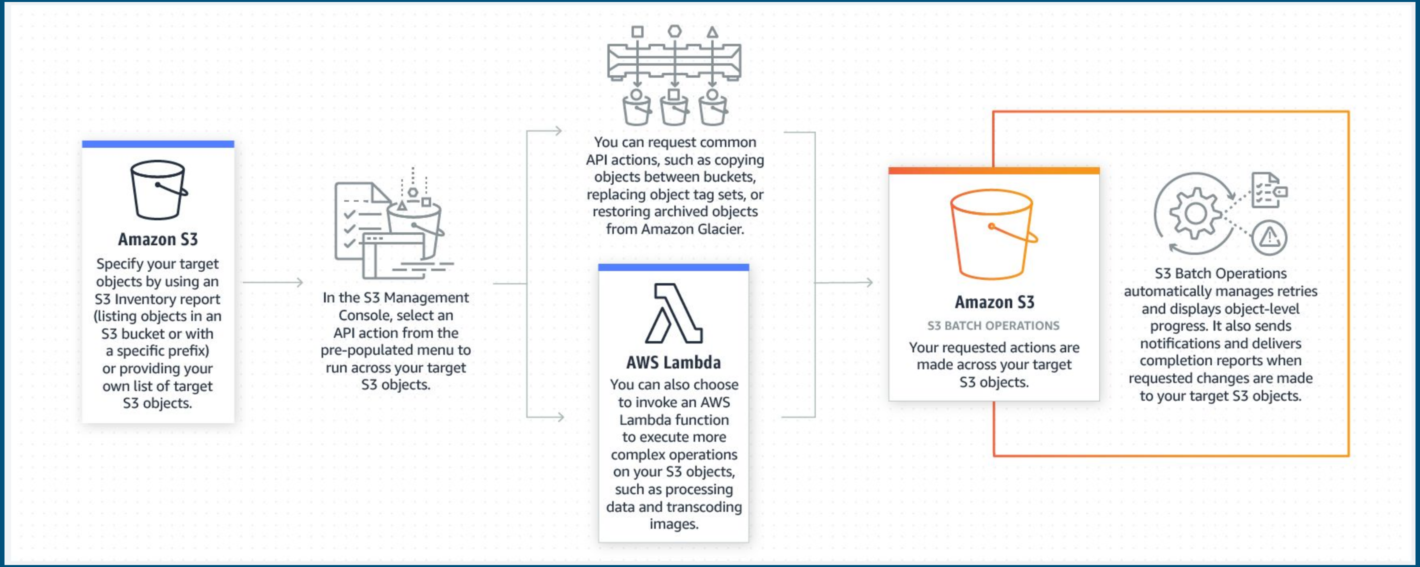
Technologies

To build this project, we used the following technologies:

1) Angular - TypeScript-based open-source web application framework, developed by the Angular Team at Google.
2) Amazon Web Services (AWS) - on-demand cloud computing platforms.
    a) Amazon S3 (simple storage service) - provides object storage through a web service (AWS) interface.
    b) Amazon DynamoDB (database) - NoSQL database service that supports key-value and document data structures offered by AWS.
3) Travis CI - Continuous integration service used to build and test software.
4) Selenium - Automated functional testing of web applications.

Back-end Technology

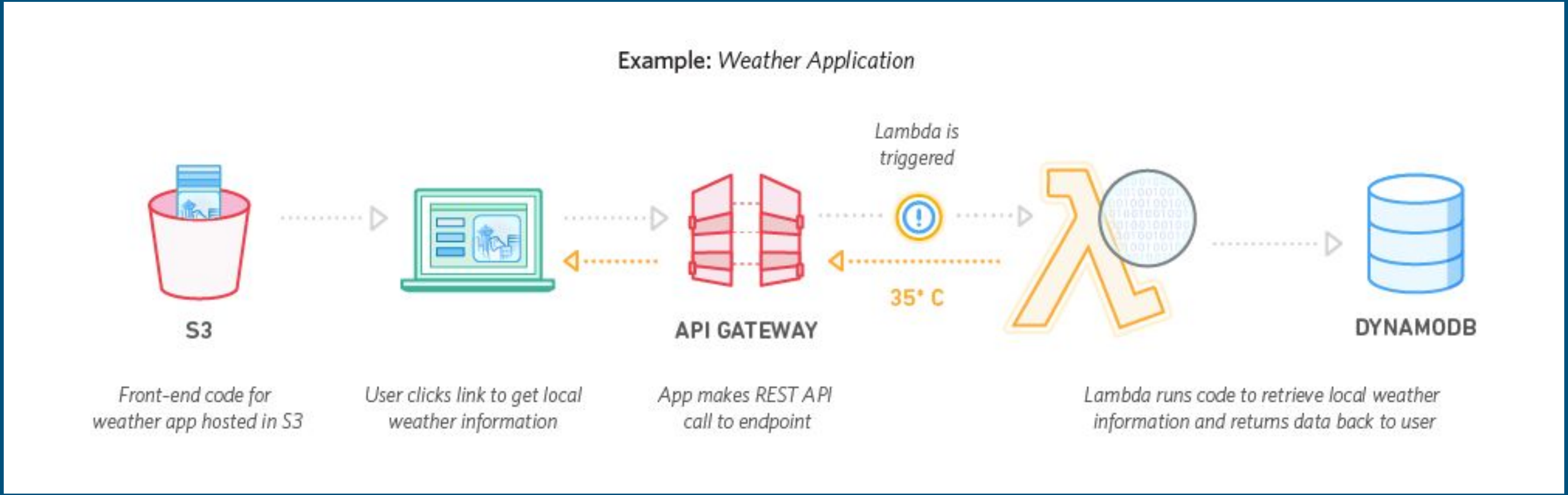
Amazon S3 - How it works


Amazon S3
Specify your target objects by using an S3 Inventory report (listing objects in an S3 bucket or with a specific prefix) or providing your own list of target S3 objects.
In the S3 Management Console, select an API action from the pre-populated menu to run across your target S3 objects.
You can request common API actions, such as copying objects between buckets, replacing object tag sets, or restoring archived objects from Amazon Glacier.
AWS Lambda
You can also choose to invoke an AWS Lambda function to execute more complex operations on your S3 objects, such as processing data and transcoding images.
Amazon S3
S3 BATCH OPERATIONS
Your requested actions are made across your target S3 objects.
S3 Batch Operations automatically manages retries and displays object-level progress. It also sends notifications and delivers completion reports when requested changes are made to your target S3 objects.

Reference: https://aws.amazon.com/s3/?hp=tile&so-exp=below

Back-end Technology

Amazon DynamoDB - How it works


Example: *Weather Application*
Lambda is triggered
35° C
S3
API GATEWAY
DYNAMODB
Front-end code for weather app hosted in S3
User clicks link to get local weather information
App makes REST API call to endpoint
Lambda runs code to retrieve local weather information and returns data back to user

Reference: https://aws.amazon.com/dynamodb/?hp=tile&so-exp=below

System Architecture & Design

Angular is a platform and framework for building client/web applications in HTML and TypeScript. It implements core and optional functionality as a set of TypeScript libraries that are imported into the application.

System Architecture & Design

The basic building blocks of an Angular application are NgModules, which provide a compilation context for components. NgModules collect related code into functional sets; an Angular app is defined by a set of NgModules.

System Architecture & Design

An app always has at least a root module that enables bootstrapping, and typically has many more feature modules.

System Architecture & Design

An app always has at least a root module that enables bootstrapping, and typically has many more feature modules.

- Components define views, which are sets of screen elements that Angular can choose among and modify according to the program logic and data.

System Architecture & Design

An app always has at least a root module that enables bootstrapping, and typically has many more feature modules.

- Components define views, which are sets of screen elements that Angular can choose among and modify according to the program logic and data.
- Components use services, which provide specific functionality not directly related to views. Service providers can be injected into components as dependencies, making the code modular, reusable, and efficient.

System Architecture & Design

Both components and services are simply classes, with decorators that mark their type and provide metadata that tells Angular how to use them.

System Architecture & Design

Both components and services are simply classes, with decorators that mark their type and provide metadata that tells Angular how to use them.

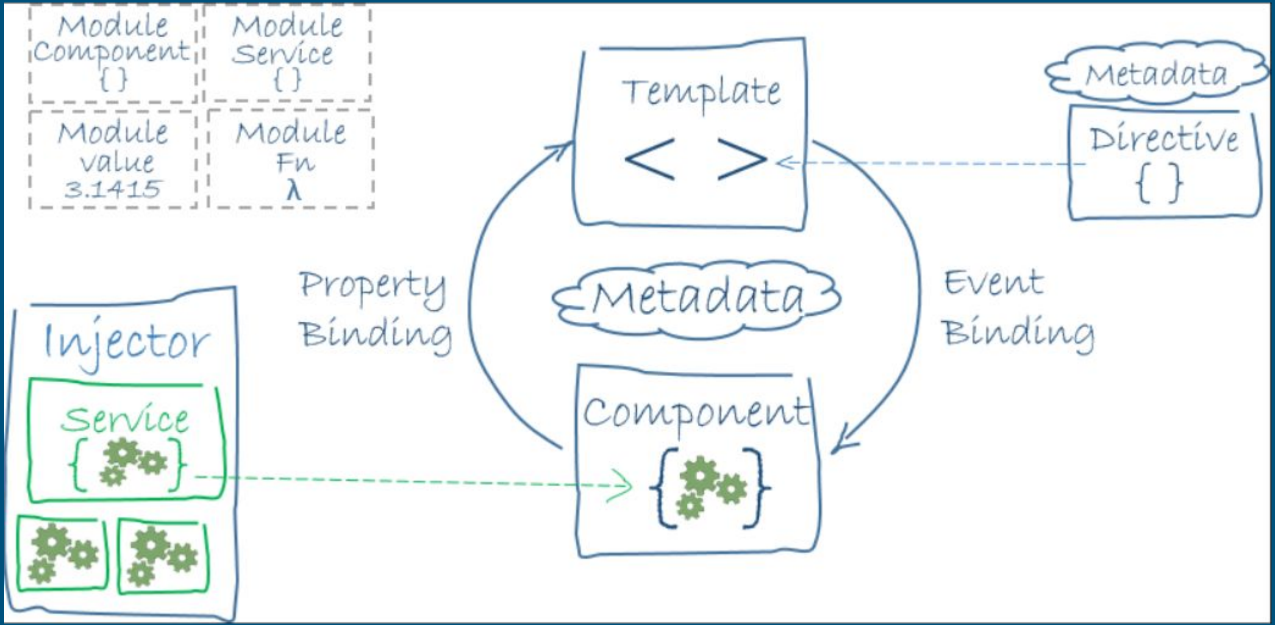
- The metadata for a component class associates it with a template that defines a view. A template combines ordinary HTML with Angular directives and binding markup that allow Angular to modify the HTML before rendering it for display.

System Architecture & Design

Both components and services are simply classes, with decorators that mark their type and provide metadata that tells Angular how to use them.

- The metadata for a component class associates it with a template that defines a view. A template combines ordinary HTML with Angular directives and binding markup that allow Angular to modify the HTML before rendering it for display.
- The metadata for a service class provides the information Angular needs to make it available to components through dependency injection (DI).

System Architecture & Design



Angular architectural reference: https://angular.io/guide/architecture

Quality Assurance & Testing

Our primary quality assurance and testing tools were Travis CI and Selenium.

Quality Assurance & Testing

Our primary quality assurance and testing tools were Travis CI and Selenium.

- Travis CI was used to continuously integrate our project throughout the development process.

Quality Assurance & Testing

Our primary quality assurance and testing tools were Travis CI and Selenium.

- Travis CI was used to continuously integrate our project throughout the development process.
    - All development merges required successful build testing before approval.

Quality Assurance & Testing

Our primary quality assurance and testing tools were Travis CI and Selenium.

- Travis CI was used to continuously integrate our project throughout the development process.
  - All development merges required successful build testing before approval.
- Selenium side tests were used to ensure all features of the web application were navigable and functional.

Quality Assurance & Testing

Our primary quality assurance and testing tools were Travis CI and Selenium.

- Travis CI was used to continuously integrate our project throughout the development process.
  - All development merges required successful build testing before approval.
- Selenium side tests were used to ensure all features of the web application were navigable and functional.
- Additionally, as required, two user evaluations were conducted to gain insight into the web application’s functionality.