StartupsNYC

Startup EcoSystem Mapping

Céline Gauchey
Department of Computer Science
Columbia University
New York, NY
cg2898@columbia.edu

Sam Friedman
Department of Computer Science
Columbia University
New York, NY
smf2240@columbia.edu

Lexa Huang
Department of Computer Science
Columbia University
New York, NY
sh4350@columbia.edu

ABSTRACT

The absence of an efficient online platform for visualizing and analyzing the startup ecosystem poses challenges for entrepreneurs, investors, and policymakers in making well-informed decisions and fostering collaboration. This deficiency impedes the identification of opportunities within the startup community.

Recognizing the pivotal role of the startup ecosystem in economic growth and innovation, we present a practical solution for stakeholders in NYC. StartupNYC, a centralized cloud-based platform, aims to empower users by providing effective tools for ecosystem mapping, investment facilitation, and informed decision-making.

In the landscape of accessing startup information, entrepreneurs and investors commonly resort to traditional methods like word-of-mouth referrals and networking events. While these face-to-face interactions are crucial for building trust within the startup community, online avenues are equally vital for expanding data accessibility and knowledge.

Our innovation lies in bridging the gap between these existing online and offline channels. Our platform enhances online channels with improved visualization and interactive mapping features, incorporating the collaborative and interpersonal elements of offline methods.

This holistic approach sets us apart, providing a user-friendly, all-encompassing solution for the startup ecosystem.

StartupNYC's primary features include web scraping of startup, investor, and relevant news, enabling users to search by category or term. The platform targets a diverse audience, including entrepreneurs seeking investment, angel investors, venture capitalists, incubators, accelerators, policymakers, and anyone interested in understanding or contributing to the NYC startup ecosystem.

December, 2023, New York, NY USA

978-1-4503-0000-0/18/06...$15.00

CCS CONCEPTS

Cloud Computing • Software Engineering • Security and privacy • Database Systems

KEYWORDS

Amazon Web Services, ReactJS, Cloud Computing

ACM Reference format:

Celine Gauchey, Samuel Friedman and Lexa Huang. 2023. StartupsNYC., *New York, NY, USA, 7 pages.*

1 Problem Statement

The lack of a comprehensive, user-friendly online platform for visualizing and analyzing the startup ecosystem in NYC hinders the ability of entrepreneurs, investors, and policymakers to make informed decisions, identify opportunities, and foster collaboration within the startup community.

1.1 Existing Solutions

In the realm of accessing startup information and the broader entrepreneurial ecosystem, there are existing online and offline avenues. Entrepreneurs and investors often rely on traditional offline methods, such as word-of-mouth referrals, networking events, and industry-specific gatherings, to discover and connect with startups. These face-to-face interactions are pivotal in building valuable connections and nurturing trust within the startup community.

On the online front, there are startup aggregators like Crunchbase, Betalist, AngelList, and StartupLister. These platforms, while useful, have limitations, often focusing on specific aspects of the startup landscape and offering only basic visualization tools.

1.2 Current Solution Limitations

The existing solutions in the market, particularly Crunchbase, present several limitations that our app aims to address. Firstly, Crunchbase restricts the ability to save information about startups and investors to its premium subscribers. This paywall limits access to vital data for many users. In contrast, our app provides the flexibility to save this information without the necessity of a premium subscription.

Secondly, Crunchbase offers a rather static experience with no insights or visual representations. Users are unable to view any analytical charts or graphics, which are crucial for understanding trends and making informed decisions. Our app stands out by offering a user-friendly interface equipped with insightful charts and graphical representations, enhancing the user's experience and understanding of the data.

Moreover, Crunchbase functions primarily as a database with search capabilities but lacks features to display top startups and investors. It does not facilitate an easy discovery of leading entities in the startup ecosystem. Our app addresses this gap by prominently featuring the “top” startups and “top” investors, thereby guiding users towards significant and trending entities in the market.

Another significant limitation of Crunchbase is its inadequate coverage of the latest news in the startup ecosystem. Staying updated with recent developments is crucial for stakeholders in this dynamic field. Our app overcomes this limitation by integrating the latest news and updates, ensuring users have access to timely and relevant information.

Lastly, the user interface (UI) of Crunchbase is not particularly user-friendly. It often appears cluttered, with an overwhelming amount of information crammed into a small space, making it challenging to read and navigate. In contrast, our app is designed with a clear, intuitive UI, prioritizing ease of use and readability. This design approach ensures that users can effortlessly access and understand the information they need.

In summary, our app addresses the critical limitations of current market solutions by offering enhanced data access, user-friendly visualization, comprehensive information on leading entities, updated news coverage, and an intuitive user interface, thereby providing a more holistic and satisfying user experience.

2 Architecture Design

2.1 AWS Services

Our website’s backend was entirely hosted using AWS services. Utilizing AWS services was, for us, a move away from building a monolithic application in favor of a front end that calls various specialized microservices.

2.1.1 API Gateway

We used a REST API implemented through API Gateway to facilitate the connection between our React front end and server-less backend, composed mainly of Lambda functions that communicate with other AWS services.

2.1.2 Lambda

The system architecture for managing startup and investor data leverages AWS Lambda functions to facilitate various data operations. For startups, three Lambda functions are deployed: 'index-startups,' 'lookup-startup,' and 'lookup-all-startups.' The 'index-startups' function is responsible for indexing startup data into the 'companies' index of the OpenSearch instance 'startups.' This indexing includes attributes like the number of employees, industry classification, total funding acquired, and geographical region. The 'lookup-startup' function integrates with the '/companies' API method, processing search requests from users. This function handles various filters and search

queries, ultimately producing identifiers (IDs) of startups that match the applied filters. These IDs are then utilized to fetch detailed profiles from DynamoDB, which are subsequently displayed to the user. Meanwhile, the 'lookup-all-startups' function is designed to retrieve a comprehensive list of all startups in the database, devoid of any filtering, and is integrated with the /all-companies endpoint.

Similarly, for investors, analogous Lambda functions are implemented: 'index-investors,' 'lookup-investor,' and 'lookup-all-investors.' These functions mirror the roles of their startup counterparts, tailored to the specific data structure and requirements of investor information, and are linked to the /investors and /all-investors endpoints.

The utilization of AWS Lambda in this context presents several advantages. Primarily, Lambda offers a serverless computing framework, which allows for efficient handling of data processing tasks without the need for dedicated server management. This results in a system that is both cost-effective and scalable, as Lambda functions are executed in response to events and automatically scale with the number of requests. Furthermore, the integration of Lambda with other AWS services like OpenSearch and DynamoDB provides a seamless and streamlined workflow, enhancing the system's overall efficiency and reliability.

In the implementation of the news feature within our system, two AWS Lambda functions, 'news' and 'lookup-all-news,' play pivotal roles. The 'news' function is tasked with scraping news articles from TechCrunch, focusing on key attributes such as 'id,' 'title,' 'author,' 'published_date,' 'summary,' 'article_text,' 'image_url,' and 'article_link.' These attributes are systematically stored in a DynamoDB table named 'news.' This function is triggered daily by an Amazon EventBridge rule, ensuring a regular update of the news content.

The rationale behind storing scraped news in DynamoDB, as opposed to directly presenting it to users, stems from a strategic decision to optimize performance and scalability. Directly scraping news for each user request would significantly prolong the response time, given the inherent latency in web scraping processes. By pre-scraping and storing news articles in DynamoDB, the system circumvents the need for repetitive scraping, thereby substantially speeding up the process of news delivery to users. Furthermore, this approach allows for efficient scaling, as DynamoDB can handle a high throughput of read requests, ensuring that the news content is swiftly and reliably delivered to a large number of users concurrently.

The 'lookup-all-news' function is seamlessly integrated with the '/all-news' API endpoint. Its primary function is to retrieve all the news items from the DynamoDB table and return them to the frontend. This enables the news page to display the latest news content to users. The integration of this function with the DynamoDB table ensures that users receive a comprehensive and up-to-date overview of news articles, enhancing their experience on the platform.

2.1.3 **DynamoDB**

We utilized DynamoDB tables to save all of our website's data, including startups, investors, and news, as well as user information and data selections saved by each user.

Among our many architecture considerations for this project was the type of database we wanted to utilize. Given that user tables, including both user information and saved items are relationally schematized, it would have been practical to use a relational database for this part of the project. However, two factors led us to use noSQL DynamoDB tables.

First, through inspection of the AWS options available to us, we came to understand that Amazon's Relational Database Service (RDS) was prohibitively expensive for use in this project. With that in mind, our second consideration had more weight. The other (non-user) data sources, although mostly structured, have the capacity for adding different features depending on the source from which the data was collected. Such made DynamoDB a very practical use case. Since we intended to implement these data stores using DynamoDB, we felt it made sense to store all of our data in one place to simplify the workflow. Had cost not been an issue for us, we would have preferred to separate out our data into two databases, as we might expect a performance boost using RDS for CRUD interactions.

2.1.4 **OpenSearch**

In our integrated data management system, a singular OpenSearch instance named 'startups' is utilized, within which two distinct indexes are maintained: one for companies and the other for investors. The companies index in this instance comprehensively catalogs various attributes of startups, including the number of employees, industry classification, total funding acquired, and geographical region. Concurrently, the investors index within the same OpenSearch instance systematically records pertinent data on investors, encompassing details such as

their total funding capacity, industry focus, type of investor, and regional presence.

When users interact with the system to filter and view specific profiles, they are able to apply diverse filters. For instance, in the context of startups, users can filter based on the size of the company (number of employees) or the industry sector. This initiates a targeted search within the 'companies' index of the OpenSearch instance to pinpoint startups matching these criteria. A similar mechanism is in place for the investors index, where user-applied filters instigate a search process to identify relevant investors.

The primary output from these OpenSearch queries are identifiers (IDs) that correspond to either startups or investors, contingent upon the user's applied filters. These IDs are subsequently used to query DynamoDB, which is tasked with retrieving detailed profiles of the startups and investors identified. The information fetched from DynamoDB is then displayed to the user.

This methodical approach, deploying OpenSearch for initial filtering within the 'startups' instance and then utilizing DynamoDB for the retrieval of detailed data, capitalizes on the strengths of both systems. OpenSearch, known for its potent search functionality, adeptly handles complex queries across large datasets. Meanwhile, DynamoDB brings its high-performance database characteristics to the fore, ensuring rapid and efficient access to detailed data profiles, thus offering a cohesive and effective data management solution.

2.1.5 **Lex**

In addition to the predefined filter options on the startups and investors page, a more flexible and interactive search feature is implemented using a search bar. This search bar is powered by an Amazon Lex bot, named 'searchQueryBot,' designed to process and interpret free-form user queries. The bot is built around a single custom intent, aptly named 'searchIntent.' It has been trained on a variety of sample utterances that reflect common user queries, such as 'show me companies with 1-10 employees in New York' or 'angel investors in Norway.'

When a user inputs a query into the search bar and initiates a search, Amazon Lex takes the lead in handling this query. The primary function of Lex in this context is to parse and disambiguate the user's input, effectively extracting key search terms and phrases. These extracted keywords are then passed on to an OpenSearch search query, which is tasked with retrieving relevant startups or investors that align with the user's search intent.

The utilization of Amazon Lex in this architecture offers several advantages. Firstly, Lex's natural language understanding (NLU) capabilities allow it to effectively interpret user queries, even when they are phrased in conversational language. This makes the search experience more intuitive and user-friendly, as it accommodates a wide range of query formulations. Secondly, Lex's integration with OpenSearch enhances the overall search functionality, bridging the gap between natural language queries and structured data searches. By leveraging Lex's ability to discern key elements from user input, the system can conduct more targeted and relevant searches in OpenSearch, thereby improving the accuracy and efficiency of the search results.

2.1.6 **EventBridge**

In order to ensure the continuous provision of updated and fresh news content within our system, we have implemented a scheduled event mechanism using Amazon EventBridge. This scheduling is encapsulated within a construct named 'news_trigger.' The primary role of this event trigger is to activate an AWS Lambda function “news” on a daily basis. Once invoked, this Lambda function is responsible for executing a routine that scrapes the latest news content from designated sources. Subsequently, the scraped news data is systematically stored in our DynamoDB database, thereby keeping the news content accessible to users current and refreshed each day.

Another task we implemented relying on EventBridge was the automatic scheduling of daily newsletter emails to our users. Every day, after the first task has run, another Lambda function is triggered to aggregate all of the titles, summaries, and links of relevant articles and send each user – who had signed up for the Newsletter upon registration – an email with results for the day.

The choice of Amazon EventBridge for these tasks is underpinned by several strategic advantages. Firstly, EventBridge offers a highly reliable and scalable event-driven architecture, which is crucial for ensuring consistent and timely execution of the news scraping routine. Its ability to trigger Lambda functions based on defined schedules allows for the automation of the news update process, thereby eliminating manual intervention and ensuring regular content updates. Secondly, the integration of EventBridge with AWS Lambda and DynamoDB facilitates a seamless workflow, where the entire process of news acquisition, processing, and storage is streamlined and efficiently managed. This integration not only simplifies the architecture but also enhances the robustness and maintainability of the system.

2.1.7 Simple Email Service (SES)

In order to ensure that our system would be able to scale to thousands of users, we applied and were approved for permission to have a production SES account, outside of the sandbox. As such, we are able to send 50,000 emails per day currently, with the possibility of scaling up with demand. SES is a very approachable method for sending out newsletters and interacting with our users.

2.2 Code Structure

React is the modern gold standard framework for front-end development. As such, we opted to use a React frontend, which we linked to the AWS backend API calls via Axios, a promise-based HTTP client for Node.js.

2.2.1 Home

The application features main pages including home, dashboard, industries, companies, investors, news, and a sign-in option. The user is able to view the application whether or not they are logged in. For non-signed-in users, the home page replaces the personal dashboard, presenting an engaging and informative interface with captivating graphics. This generic home page outlines what the application offers and provides options to log in or sign up, inviting users to access personalized features and a more tailored experience.

2.2.2 Dashboard

The dashboard of the application is designed to enhance user engagement and data visualization. It prominently displays the first five startups and investors saved by the user, with an option to view more through a "see more saved startups" or "see more saved investors" feature. Clicking these options opens a dialog showcasing all saved entities, rather than just the initial five. Additionally, the dashboard incorporates two pie charts for a graphical representation of the industries of both startups and investors saved by the user, providing a clear and concise visual summary of their saved preferences.

2.2.3 Industries

The Industries Page of the application is designed for dynamic and informative user interaction. It initially showcases the top five startups and investors, with names, descriptions, and locations clearly visible. A "see more" feature allows users to access a comprehensive list of all startups or investors in a dialog format. In the Top Startups section, users can view the names and descriptions of the first five startups, with a "see more top startups" option revealing a comprehensive list. Similarly, the Top Investors section displays the names and locations of the top five investors, offering a "see more top investors" feature for extended exploration. Additionally, when searching for a specific industry, the page dynamically updates to display the top five startups and investors in that sector, with expanded lists accessible via the same "see more" functionality. The page culminates with a news section at the bottom, presenting the latest articles with images, headlines, and publication dates, and links to full articles for in-depth reading.

2.2.4 Companies

The Companies page provides a user-friendly interface for viewing and managing information on various firms. It features a search bar with filters for industry, employee count, region, and funding, allowing for targeted queries. Displayed in a tabular format, the page lists companies with pertinent details such as name, industry, location, and a brief description. Users can view a default list of five companies per page, with navigational options to scroll through additional entries. This structured presentation aids users in quickly finding and saving relevant company information for future reference.

Upon selecting a company from the list, the interface presents a detailed dialog box providing an enriched set of data about the chosen entity. This popup contains comprehensive information including the company's name, the sectors it operates in, location, total funding received, a succinct description, and a direct link to the company's website. This feature allows users to delve deeper into the specifics of each company, facilitating a more informed and efficient browsing experience.

When a user selects one or more companies via checkboxes, the app records these selections. The saved data is then prominently displayed in the 'Saved Startups' section on the user's dashboard. This feature allows for easy retrieval and reference, enabling users to curate a personalized list of companies they are interested in tracking or analyzing further.

2.2.5 Investors

The Investors page of the application is a user-centric interface similar to the Companies page. It provides a list of investors, categorized by name, investor type, location, and total funding. It features a search function with multiple filters to streamline user queries, including a search bar with filters for investor type, region, and funding. When a user selects an investor, a detailed modal appears, offering additional information including name, investor type, location, total funding in USD, and the date it was founded on. Users can select various investors through checkboxes

to save their information. Selections made here can be saved and are later accessible in the “Saved Investors” section of the user's dashboard, allowing users to easily revisit their selections and analyze their investment profiles.

2.2.6 News

The News page is tailored to deliver articles pertinent to the startup ecosystem, showcasing a plethora of articles with accompanying images, headlines, summaries, authors, and publication dates. This design not only engages users with current events and analyses but also serves as a gateway, redirecting them to the complete articles for an in-depth read, thus keeping the entrepreneurial community informed and connected.

3 Code Organization

The React repository's file structure is modular, divided into logical sections that reflect the application's architecture. At the top level, the 'src' folder holds the 'components' directory, which is further categorized into feature-specific subdirectories such as 'companies', 'dashboard', ‘home’, ‘industries’, ‘investors’ and ‘new’. Each subdirectory contains JavaScript files that define the components and their functionality related to that feature. In addition, we have created a ‘common’ folder for reusable components. A common folder for reusable components in a React project centralizes the core, shared elements of the application, fostering reusability and ensuring a DRY (Don't Repeat Yourself) codebase which simplifies maintenance and scalability. Each subfolder encapsulates the logic and UI for its domain, ensuring that components remain organized and the codebase is easily navigable. This hierarchical organization facilitates maintenance and scalability, allowing us to locate and manage specific parts of the application efficiently.

3.1 Git Repository

We utilized Git and GitHub for version control and collaboration in our frontend repository, allowing the team to work concurrently on different features without overlap, thanks to multiple branches. Our use of Git and collaborative practices on GitHub, including thorough peer reviews and consistent integration of new code, resulted in a development process free from major merge conflicts or other significant issues.

Our GitHub repository can be found here:
https://github.com/cgauchey/cloud-final-proj/

3.2 CI/CD Pipeline

In order to enable best practices for continuous integration and continuous deployment, we set up a CodeBuild project that is linked to our Git repository via a webhook. Upon committing code to the main branch, our build project detects the change and initiates the process of installing all npm packages, building our react app and cloning the build folder into our s3 bucket. The target s3 bucket has its endpoint exposed at su-nyc.s3-website-us-east-1.amazonaws.com where it is accessible to any parties online.

4 Security & Data Governance

4.1 Authentication

We opted to implement the authentication mechanism ourselves, without a managed service such as AWS Cognito. The reason for this was two-fold. First, we wanted to have maximal control over our system. Second, we felt that this choice simplified our connection with our database.

4.2 Use of 3rd Party Data

Data for our project was obtained via an API from CrunchBase and web scraping TechCrunch. Both sources allow the use of gathering data, granted reference to each respective data source is listed on our website. We stored all data in DynamoDB tables, as referenced in section 2.1.3.

5 Results

5.1 Success Criteria

Our original plan had two criteria for success:

The first success criteria was “Build an interactive map to visualize startups in New York City.”
Making it interactive is more front-end and we wanted to build a more robust backend, but we have certainly produced result-bearing tables that users can interact with and use to save state information. In addition we have add other features.

The second success criteria was “Build a system that allows for different users to create their own accounts to save relevant startup information.”
We have certainly accomplished this.

5.2 Changes to Original Plan

Our original plan was light on backend architecture, so we made sure to expand it in ways that count. We implemented

a lex bot for startup/investor data filtering based on key work searches. Similarly we use OpenSearch to select categorical data. Additionally we use EventBridge to trigger daily updates of the news articles in our databases, as well as utilize both EventBridge as well as SES to create an automated daily newsletter accessible to anyone who registers on our website. All of these features were added to the original plan in order to make a web-app that focuses heavily on cloud-computing elements as opposed to front end design.

5.3 Limitations or Bugs

Our original intention was for company and investor data to be sourced from publicly available information, government databases, startup directories, and user contributions. Three major platforms we aimed to collect data from were Crunchbase, Techcrunch, and AngelList. However, in practice collecting this data was difficult. In some cases, popular websites either deactivated their APIs or monetized them in such a way that was cost prohibitive for our group. In other cases, since our website is public-facing, data scraping other sources would have posed risk for copyright infringement. Ultimately, we had to settle with collecting our data from the CrunchBase API and through web scraping TechCrunch.

ACKNOWLEDGMENTS

We would like to thank our Professor, Sambit Sahu, for an excellent course on Cloud Computing and Big Data. We would also like to thank our Teaching Assistants who worked tirelessly to facilitate the course and answer our questions over the course of the semester.

REFERENCES

[1] https://docs.aws.amazon.com/
[2] https://docs.aws.amazon.com/amazondynamodb
[3] https://docs.aws.amazon.com/apigateway
[4] https://docs.aws.amazon.com/lambda
[5] https://docs.aws.amazon.com/opensearch-service
[6] https://docs.aws.amazon.com/lex
[7] https://docs.aws.amazon.com/eventbridge
[8] https://docs.aws.amazon.com/ses
[9] https://docs.aws.amazon.com/codebuild