Bilkent University

Department of Computer Engineering

CS 491 - Senior Design Project

Fall 2024

Project Specification Document

T2419

Bora Haliloğlu - 22101852

Burak Oruk - 22102443

Emir Tuğlu - 22003165

Gökalp Gökdoğan - 22102936

Tevfik Emre Sungur - 22102377

Table of Contents

1. Introduction

1.1. Description

SürDur is a mobile application that offers users a pleasant road trip experience by recommending personalized stopovers along the route. After selecting a destination, users can explore a range of recommended points of interest (POIs) drawn from various databases and travel blogs, guaranteeing a customized and rich trip experience. Through its social media feature, which allows users to vote on routes, follow other travelers, and discuss their travel experiences, SürDur promotes community involvement in addition to route planning. This social component improves user engagement and offers feedback loops for personalized recommendations.

In this report, we will cover the application's high-level system architecture, detailing its four main layers: Presentation, Business Logic, Data Access, and Database Layers. Also, we will discuss various functional and non-functional requirements, including user authentication, destination selection, point of interest suggestions, and social media. Additionally, we will elaborate on constraints related to development, economics, technology, and social considerations while addressing professional and ethical issues. The standards that will be used in the project are also included to guide the project’s framework. Lastly, we will discuss the feasibility of the project by providing market and academic analysis.

1.2. High Level System Architecture & Components of Proposed Solution

The following component diagram shows the architecture in terms of separate, self-contained components. When joined in a certain manner, they build the whole software, and this diagram includes those connections and components. It represents SürDur's layered architecture in a slightly more detailed way. SürDur’s layered architecture is designed to promote modularity and maintainability of the project. This will ensure modern architectural principles like modularity and separation of concerns. It contains four main layers: Presentation Layer, Business Logic Layer, Data Access Layer, and Database Layer.

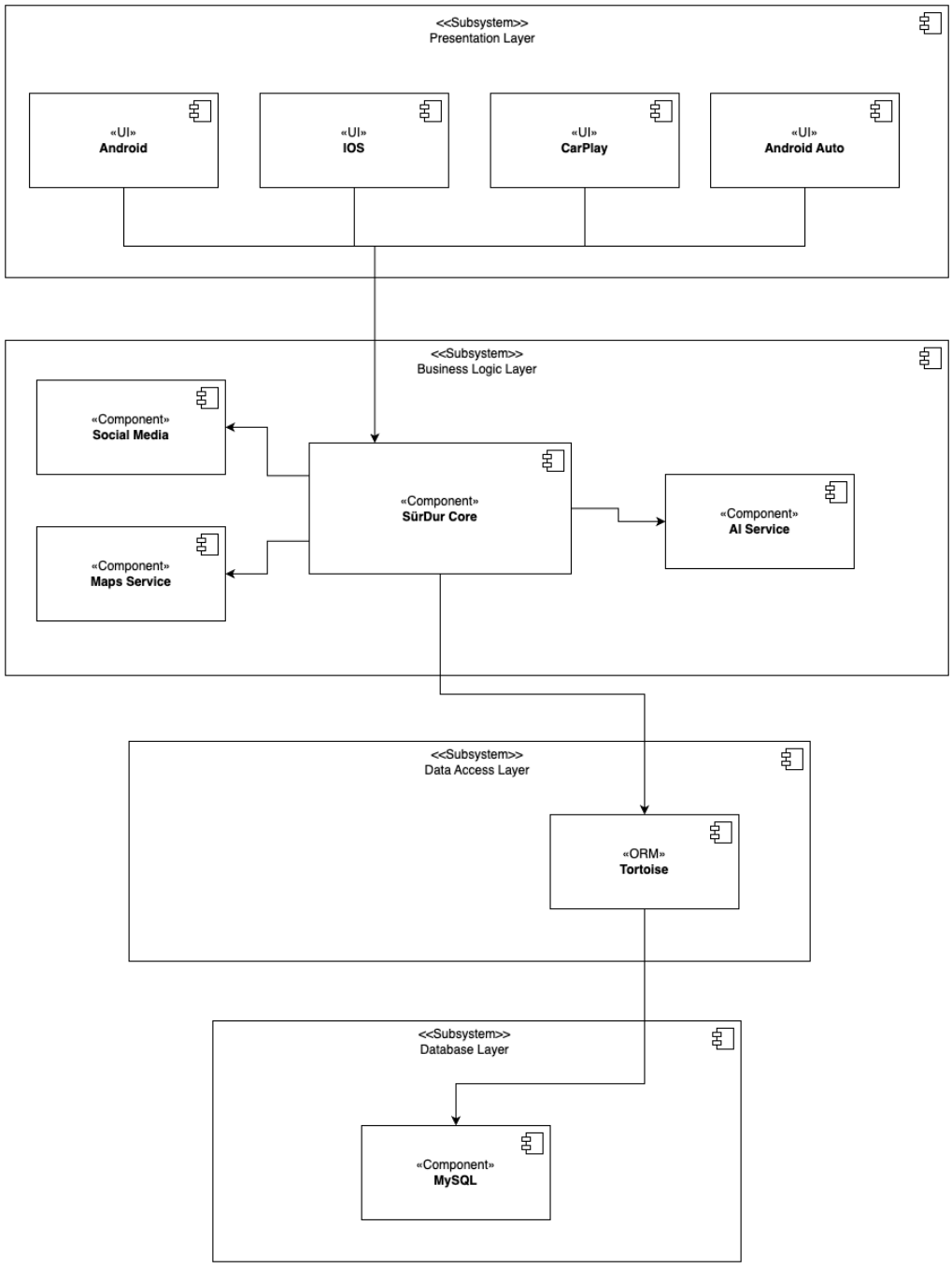

Figure 1: Component Diagram of SürDur. Link: https://shorturl.at/iVIqa

1.2.1. Presentation Layer

Presentation Layer includes four different UIs but during this project we will prioritize the development of mobile UIs(Android and iOS). This layer will essentially enable user interactions.

1.2.2. Business Logic Layer

The Business Logic Layer includes the core and supporting components like AI Service and Maps Service. Each supporting component is directly connected to SürDur's Core since the core will act as coordinator of other components. AI Service will return POI recommendations. Maps Service will be responsible for navigation and pathfinding. Social Media will be responsible for post and user interaction management. Therefore, this layer's main function is supporting the application's core business functions. It also acts as an intermediary between the presentation and data access layers.

1.2.3. Data Access Layer

The Data Access Layer manages data persistence between the Business Layer and the Database Layer with Tortoise ORM Library. It will simplify database operations.

1.2.4. Database Layer

The Database Layer uses a MySQL database which includes all the data about the users, places, posts, and route logs. It manages the data storage and retrieval. It will provide this information through Tortoise ORM in the Data Access Layer.

1.3. Constraints

This section will discuss the SürDur project's constraints in detail on aspects of development, economic, technological, social, safety, and sustainability.

1.3.1. Development Constraints

- The project app will be available for both IOS and Android.
- In a broader scope, the project's UI also will be implemented for Apple CarPlay and Android Auto.
- The mobile side of the project will be developed using React Native as it's compatible with both IOS and Android.

- The application will be developed with Python and FastAPI for the back end and React-Native for the front end.
- OpenAI API will be used to extract categories of POIs from online blogs by using language models.
- In categorizing POIs, NLP methodologies and tools will be used to help mapping many categories into pre-determined categories.
- Git and Github will be used as our version control system.
- MySQL will be used to store and access database components related to both the application engine and the user data.
- The decision engine and the database will be kept on AWS servers.
- Notion will be used to keep track of the development process.

1.3.2. Economic Constraints

- The database will be kept on AWS servers. Annual payment for reserving a micro-sized server space (1 GB of data space) from Stockholm servers requires $94 [1].
- Publishing the app on mobile platforms has two economic constraints. One is the $25 one-time registration fee on the Google Play Store (Android), and the other is the annual $99 fee on the App Store (IOS) [2].
- Frameworks and libraries that will be used to implement the project such as FastAPI, React Native, and Expo, are free to use.

1.3.3. Technological Constraints

- The application will need an internet connection for all the functionalities such as route creation, navigation, social functions, and profile operations.
- The application has to access the user's location on route creation, and navigation functions.

1.3.4. Social Constraints

- The application will allow the sharing of previously followed routes.
- The public route posts will include a title and a header which allows a detailed explanation of the route. However, there is no further text-based communication allowed on those posts.
- The posts have a voting system that shows the public appreciation of users' posts.

1.3.5. Safety Constraints

- Mobile app decisions will be made to minimize user interaction during a car ride.
- The user will be asked to make choices before starting the ride.

1.3.6. Sustainability Constraints

- Server and publishing services need to be paid annually.
- An increase in the number of users may result in database enlargement, which will result in higher server space costs.

1.4. Professional and Ethical Issues

1.4.1. Professional Issues

- Through surveys and notices end users will be informed about the project process after design changes or finalizations. Their feedback will be taken into consideration in the development process.
- The demo app will be presented to end users to get their feedback and shape the app to match the market's expectations.
- Every city and region of Turkey will be prioritized equally when a database is being constructed. However, it should be noted that the unequal distribution of human-made stops such as restaurants makes it not possible to provide the same density of solutions in all cities and regions.

1.4.2. Ethical Issues

- The geolocation of the users will not be kept or shared in any means. It will be used only in the scope of creating a route and navigation.
- Personal information such as the recommendation info of users and past trips will be kept private.
- Personal information such as name, surname, profile photo, and shared routes will be displayed to the public on the app. However, this information will be kept private outside the app display.

1.5. Standards

1.5.1. IEEE 1471

Purpose:

IEEE 1471, a software-intensive system architecture documentation standard, is ISO/IEC/IEEE 42010. It outlines developing an architecture description that offers a shared comprehension of the system's structure, functionality, and essential

characteristics [8]. We can more easily comprehend system components and their relationships thanks to IEEE 1471's assistance in clarifying the architecture. This entails outlining the architecture's background, perspectives, interested parties, and the reasoning behind essential choices [3].

Key Elements:

We document architectural choices in development; this outlines essential decisions made during the design process, supporting information, and other factors. Also, documenting architectural views in the project enables us to represent the project's physical, process, development, and logical aspects.

1.5.2. UML 2.5.1

Purpose

A widely used modeling language for describing, building, visualizing, and recording the structure and behavior of software systems is UML 2.5.1 [9]. It provides a consistent method for drawing diagrams that explain various system components. We can understandably display the system's structural and functional elements using UML. This standard facilitates the creation of models for different views (such as class, sequence, and activity diagrams), which helps with system design and communication [4].

Key Aspects

Class, Component, and Deployment aspects define the system's static structure. Use Case, Sequence, and Activity represent dynamic aspects of the system, including interactions and workflows.

1.5.3. IEEE 830

Purpose

Writing Software Requirements Specifications (SRS) is standardized by IEEE 830. It establishes a thorough framework for recording functional and non-functional requirements, guaranteeing accuracy, consistency, and comprehensiveness [10]. IEEE 830 offers a systematic style for specs reports that assists teams in organizing requirements for easy understanding and verification by stakeholders, developers, and testers. Project objectives, scope, requirements, assumptions, and restrictions are all covered in this standard [5].

Key Aspects

This standard addresses the project's background, goal, and extent. Additionally, it gives a summary of the operating environment, user attributes, and product capabilities.

1.5.4. ISO 31000

Purpose

One standard that offers recommendations for efficient risk management is ISO 31000. It aids businesses in recognizing, evaluating, and reducing risks, which enhances decision-making and reduces uncertainty [11]. This standard exemplifies proactive risk management by addressing potential project risks (technical, operational, and financial) and mitigation techniques. Risk assessment frameworks, prioritization, and controls are a few examples [6].

Key Aspects

Identifying potential risks that have an impact on the project. Assessing the impact and probability of hazards that have been discovered. establishing measures to reduce or eliminate risks. We can identify possible problems and dangers by implementing risk management.

1.5.5. IEEE Citation Style

Purpose

IEEE Citation Style is a widely standardized approach for citing sources from engineering, information technology, and allied fields. It increases the traceability and dependability of the information by ensuring that sources are consistently mentioned. IEEE Citation Style provides a uniform method of referring to external sources (including research papers, technical publications, and standards) that ensures accuracy and lucidity. When citations are appropriately formatted, readers may locate sources for further context and proof [7].

Key Aspects

References match the list of references and are numbered in brackets (e.g., [1], [2])[12]. provides comprehensive information for every source and arranges citations in numerical order.

2. Design Requirements

2.1. Functional Requirements

This section will describe the functional requirements that SürDur application should include for proper implementation.

2.1.1. Sign Up & Login

Application should:

- Allow users to log in to the application using their account information.
- Allow users to register manually by providing account details.
- Allow users to add their personal information, such as name, age, and preferences.
- Enable users to add place category preferences for road trips.

2.1.2. Destination Selection

Application should:

- Display the user's current location on an interactive map.
- Allow users to move around the interactive map.
- Provide a feature to return to the user's current position on the map.
- Allow users to select a destination location for route planning.

2.1.3. Point of Interest Suggestion

Application should:

- Display suggested points of interest (POIs) close to the given route.
- Allow users to zoom in or out to a particular region to get an adequate number of suggestions in that area.
- Allow users to filter POIs by category to display only the POIs with the selected categories.
- Display the preview information about the suggested places.
- Display detailed information about the selected place among the list of suggested places.
- Allow users to add places to the route from the given suggestions.

- Allow users to remove places from the route among all of the previously selected places.
- Reconstruct the route dynamically after each place addition and removal.
- Display the estimated time to reach the destination and the percentage of how off the new route is from the original route.
- Allow users to finalize the route with selected places and save it on the ‘Planned’ routes of the user.

2.1.4. Social Media

Application should:

- Display the shared route posts from other users on the application, with the components:
    - Starting and Destination Location
    - Interactive Map Overview of The Route
    - Title
    - Description
    - Author Username and Profile Picture
    - Upvote and downvote counts
- Allow users to display detailed description of the selected route among shared routes.
- Open the selected shared route by the user on the interactive map.
- Allow users to upvote or downvote the route post.
- Allow users to follow and unfollow other users
- Allow users to save the selected shared routes on the ‘Saved Routes’ folder of the user.
- Send users notifications when there are upvotes or downvotes on their posts.
- Send users notifications when the users that are followed share their routes.

2.1.5. Archives

Application should:

- Display all the personal routes of the user that are divided into four route categories:

    - In Draft: Routes that have not been totally completed on planning by the user and require further route planning completion from the user.
    - Planned: Routes that have been assigned as ‘completed planning’ by the user and are ready to be traveled afterward.
    - Saved: Routes from other users that have been saved on the local archive from the social media
    - Finalized: Routes that have been traveled and finished by the user in real life.
- Allow users to filter personal routes based on their category, destination location, and start location.
- Open the selected route from the archive on the interactive map for further route editing or to start the road trip.
- Allow users to delete present routes from the user archive.
- Allow users to share their routes on the social media part of the application.
- Allow users to flag certain routes for further distinction from other routes.

2.1.6. Live Route Navigation

Application should:

- Allow users to start live navigation throughout the selected route.
- Perform real-time navigation throughout the road trip.
- Display real-time road directions (next maneuver’s turn and distance etc.) on the screen.
- Display the total expected time remaining.
- Allow users to select emergency 'gas station' and 'electric vehicle charging station' stops.
- Add the nearest and most convenient gas station or electric vehicle charging station dynamically into the current route if chosen.
- Adjust live directions dynamically when the driver gets off the planned route.
- Allow users to exit the live navigation of the current route.

2.1.7. Profile Editing & Setting Adjustment

Application should:

- Allow users to edit their personal account information, such as username, password, etc.
- Allow users to adjust application experience settings, such as theme, distance and speed units, notification enablement, etc.

2.2. Non-Functional Requirements

2.2.1. Usability

The experience of the usage of SürDur will be evaluated based on both application usage time, and personalized place suggestion satisfaction. Hence, SürDur must provide a user-friendly interface by including simple, easy-to-use also comprehensive components to provide a smooth state transaction from the main page to the completion of the planned route. In that sense, SürDur should provide a proper number of place suggestions along the main route in order to not exhaust the user and also to prevent a lack of suggestions. The application must also execute smoothly for both mobile devices (including Android and iOS) and CarPlay devices.

2.2.2. Reliability

SürDur must provide an uninterrupted navigation performance throughout the road trip. Hence, the application should handle network errors by caching the map information on the local machine, and continue to serve navigation, which is independent from the network. Also, the application should include navigation services in the case of a failed satellite connection, and inform the driver about the issue safely. The application should continue its live navigation services in any possible server downtime coming from maintenance and update processes. Other than error handling criteria, the application should store backup of the main database in the case of system crash and information loss.

2.2.3. Performance

SürDur must have an advanced data-retrieval service that would minimize the response time in each route planning suggestion. The application should cache the most related places from the enormous database of place information based on the location, relevance and personal choices of the user. Other than data retrieval performance, live navigation should have minimized lagging for the driver to have a real-time driving experience for both satisfaction and safety.

2.2.4. Supportability

SürDur must have a globally compatible service and database structure. The development environment should allow the addition or removal of microservices that alter the place of information on the database without interrupting the main functionality of the application. Also, the development environment should include environment containerization services, such as Docker, to easily deploy new libraries and configurations globally with one main structure. Other than a compatible development environment, the system should include logging services to keep track of the occurring events regularly, including errors and warnings.

2.2.5. Scalability

SürDur must handle enormous workloads on large-scale user scenarios by considering storage space, user request overloading, and performance efficiency. Hence, the application should include fast and asynchronous backend services, information storage optimization, and efficient caching services. Other than large-scale user demand, SürDur also should consider vast amounts of place data, and how to store them and retrieve them efficiently. This issue also shows the importance of information storage optimization and efficient caching services.

3. Feasibility Discussions

In this part of the report, we will analyze the feasibility of the project from the market & competitive, and academic perspectives. First, we will analyze other applications in the market that solve the same problem with our application. The remarkable points of each of our competitors will be discussed. Later, we will summarize the functionality provided in our

application and in our competitors in a table. Lastly, we will discuss the academic analysis of our project.

3.1. Market & Competitor Analysis

3.1.1. Roadtrippers

- Available in only the USA, Australia, Canada, and New Zealand.
- Creating road trips is free, but navigating along them requires a premium.
- Has some pre-created road trip suggestions.
- Users can choose what categories of recommendations they want.
- There is a map-saving option for offline usage.
- Has a chatbot that generates road trips by asking questions as a premium feature.
- Has 1M+ downloads.

3.1.2. Roadie

- Has no navigation option, it exports the route to Google Maps for navigation.
- Locations can be searched by category.
- Waypoints can be added, deleted, and reordered.
- Users can save routes, but there is no sharing option.
- Free mode is restricted.
- Has 100K+ downloads.

3.1.3. Sygic Travel

- Has no navigation option, it exports the route to Google Maps for navigation.
- Dynamically retrieves locations as the user moves on the map.
- Has offline maps mode.
- Poor UI/UX.
- Has 1M downloads.

3.1.4. Google Maps

- Has an explore feature which shows all locations by category as the user moves on the map.
- Has search along the route feature. Locations are retrieved by search keywords, which is not useful.

3.1.5. Yandex Maps

- Has a separate explore option which shows locations of selected categories as the user moves on the map.
- The explore option is separated from the route option, suggested locations are not shown on the route.

3.2. Comparison with Competitors

	SürDur	Roadtrippers	Roadie	Sygic Travel	Google Maps	Yandex Maps
Navigation	+	+	-	-	+	+
Displaying the POIs and the route together on a map	+	+	+	+	+	-
Retrieving POIs from multiple sources	+	?	?	?	-	-
Personalized POI Recommendations	+	-	-	-	+	+
Sharing routes with friends	+	-	-	+	+	+
Has a chatbot assistant	-	+	-	-	-	-
Available on Türkiye	+	-	+	+	+	+

3.3. Academic Analysis

In this part, we will discuss the requirements of the project from an academic perspective. We performed a rigorous literature review to find solutions to the main problems of our project. In the subsections, we will provide information about the problems and the solutions that we designed by reviewing academic resources.

3.3.1. AI-Powered Personalized Recommendation

To be able to provide personalized POI recommendations, we will use the “Content-based Filtering” recommendation technique which is a sub-area of the Machine Learning field [13]. We are planning to create a preference vector for each user

containing the user's preference weights for a set of features that are associated with the locations. Preference weights of users will be updated by their previous choices and their interactions in the social media part of our application. We will also create weights for the same features for each POI by using the data we retrieve from various APIs. Our recommendation algorithm will recommend to user POIs whose weights align with the weights in the user's preference vector. In this way, we will be able to provide personalized recommendations to our users.

3.3.2. Data Retrieval Performance

Since we are going to provide users recommendations from a giant set of POIs, the performance of the data retrieval process is essential for our project. To be able to retrieve data in an acceptable time period, we are planning to use a Relational Database Management System (RDBMS). Also, we will create indexes for frequently queried columns in the database. Indexing will help us significantly reduce the query execution times by allowing the database to locate and retrieve rows efficiently without scanning the entire dataset. Additionally, the application will cache the most-relatable and most-interacted places into the local memory. Caching will also help us reduce the execution time by eliminating unnecessary data retrieval requests from the database. Lastly, we are planning to use query optimization techniques to further reduce the time needed for data retrieval.

3.3.3. Categorization of POIs

We will retrieve the POI information from different API providers. These providers have similar but not exactly the same categories for POIs. For example, one provider might use "Cafes" while another uses "Coffee Shops" for the same type of location. To prevent issues from such misalignments, we will create our own categories and map each category in the API providers to a category in our list. To automate this process, we will use the k-means algorithm to group similar categories into a single one. In this way, we will be able to classify POIs according to our own categories.

4. Glossary

UI: User Interface.

POI: Point of Interest. Locations to be recommended to the user.

API: Application Programming Interface. An interface that a software provides for the use of another software.

5. References

[1] Amazon Web Services, "Amazon RDS for MySQL pricing," [Online]. Available: https://aws.amazon.com/tr/rds/mysql/pricing/?pg=pr&loc=2. [Accessed: Nov. 19, 2024].

[2] Sphinx Solution, "Cost to put an app on the App Store," [Online]. Available: https://www.sphinx-solution.com/blog/cost-to-put-an-app-on-the-app-store/. [Accessed: Nov. 19, 2024].

[3] IEEE, "IEEE Standard 1471: Recommended practice for architectural description of software-intensive systems," [Online]. Available: https://standards.ieee.org/ieee/1471/2187/. [Accessed: Nov. 19, 2024].

[4] Object Management Group, "Unified Modeling Language (UML), version 2.5.1," [Online]. Available: https://www.omg.org/spec/UML/2.5.1/About-UML/. [Accessed: Nov. 19, 2024].

[5] G. Booch, J. Rumbaugh, and I. Jacobson, "The Unified Modeling Language user guide," IEEE, 1999. [Online]. Available: https://ieeexplore.ieee.org/document/720574. [Accessed: Nov. 19, 2024].

[6] International Organization for Standardization, "ISO 31000: Risk management," [Online]. Available: https://www.iso.org/iso-31000-risk-management.html/. [Accessed: Nov. 19, 2024].

[7] New Jersey Institute of Technology, "IEEE citation style guide," [Online]. Available: https://researchguides.njit.edu/ieee-citation/ieeereferencing/. [Accessed: Nov. 19, 2024].

[8] W. Pree, "Design patterns for object-oriented software development," IEEE, 1995. [Online]. Available: https://ieeexplore.ieee.org/document/875998/. [Accessed: Nov. 19, 2024].

[9] UML Diagrams, "UML 2.5 diagrams overview," [Online]. Available: https://www.uml-diagrams.org/uml-25-diagrams.html. [Accessed: Nov. 19, 2024].

[10] IEEE, "IEEE Standard 830: Recommended practice for software requirements specifications," [Online]. Available: https://standards.ieee.org/ieee/830/1222/. [Accessed: Nov. 19, 2024].

[11] International Organization for Standardization, "ISO 9001: Quality management systems," [Online]. Available: https://scc.isolutions.iso.org/obp/ui#iso:pub:PUB100464. [Accessed: Nov. 19, 2024].

[12] George Mason University, "IEEE style citation guide," [Online]. Available: https://infoguides.gmu.edu/ieee_style#s-lg-box-29326431. [Accessed: Nov. 19, 2024].

[13] “Content-based Filtering | Recommendation Systems,” Google Developers. https://developers.google.com/machine-learning/recommendation/content-based/basics. [Accessed: Nov. 21, 2024].