Bilkent University
Department of Computer Engineering

Senior Design Project

T2426 - CheckMate

Detailed Design Report

Ayberk Berat Eroğlu - 22103675

Alp Batu Aksan - 22103246

Pelin Öner - 22102409

İpek Sönmez - 22103939

Efe Tuna Can - 22102127

Supervisor: Cevdet Aykanat

1. Introduction

1.1 Purpose of the System

In an era characterized by the rapid dissemination of misinformation, ensuring the accuracy and reliability of online content has become a critical challenge. Fact-checking remains predominantly a manual process, relying on human experts to investigate claims and verify their authenticity. For instance, Meta, the parent company of Facebook and Instagram, employs human fact-checkers to assess the credibility of online content [1]. Similarly, Twitter (now X) utilizes Community Notes, a crowdsourced system that aggregates contextual insights and fact-checks based on user contributions [2]. However, these approaches are often insufficient to address the vast volume of information circulating on digital platforms, as manual fact-checking is both time-intensive and resource-demanding [3].

CheckMate, an artificial intelligence (AI) powered browser extension and mobile application designed to automate fact-checking, multimedia verification, political bias detection, and subjectivity analysis. By leveraging machine learning (ML) techniques, CheckMate aims to provide a more efficient and scalable solution for misinformation detection. As the internet continues to emerge as the dominant news source—surpassing traditional media in countries such as the United Kingdom [4]—the integration of automated verification tools across major news platforms is essential for mitigating the spread of false information.

CheckMate systematically evaluates news articles and assigns a reliability score based on multiple factors. Using natural language processing (NLP), it detects biased, subjective language and analyzes political bias, incorporating these findings into the reliability assessment. Additionally, CheckMate cross-references information with other sources via search Application Programming Interfaces (APIs) to identify confirmatory or contradictory evidence.

Metadata, including the publication source, is also analyzed to assess credibility. If a source has a history of publishing misinformation, exhibits political bias, or consistently employs subjective language, CheckMate adjusts the source's credibility score accordingly, which contributes to the overall reliability score of the article. Furthermore, CheckMate performs reverse image searches to detect inconsistencies, verifying whether visuals have been previously used in different contexts.

Upon completing its analysis, CheckMate presents the reliability score along with a rationale, highlighting factors such as high political bias, inconsistencies in information, or contradictions with reputable sources.

1.2 Design Goals

1.2.1 Usability

- The system provides an intuitive and user-friendly interface, allowing users to easily access each article's reliability score along with its rationale.
- The system displays clear visual indicators (color-coded reliability and credibility scores, political spectrum graphics) to effectively convey the reliability and political iasb of articles along with credibility of the article's source.
- The system allows users to provide feedback on reliability of the articles and any possible bugs with minimal steps.
- The system is accessible on all major Chromium browsers (Google Chrome, Opera, Microsoft Edge and Brave) and other commonly used browsers Mozilla Firefox and Safari [5]. The mobile application of CheckMate is compatible with both Android and iOS platforms optimized for responsive display across different screen sizes.

1.2.2 Reliability

- The system ensures accuracy and consistency in reliability scoring and political bias classification by continuously updating the reference database with credible sources.
- User feedback is incorporated into the system to continuously improve the reference database and refine classification and scoring models over time.

1.2.3 Transparency

- The system provides clear justifications for reliability scores, detailing contributing factors such as subjectivity and political biases, contradictory evidence, and source credibility.
- The system allows users to review and contest assessments through a feedback mechanism, fostering accountability and continuous improvement.

1.2.4 Performance

- The system parallelizes the similar article search to improve efficiency and reduce latency.

- If an article from the same news source has already been analyzed by another user, the system retrieves its reliability score analysis directly from the database, minimizing redundant processing and ensuring a seamless user experience.
- The system optimizes interactions with APIs to minimize response times and improve overall system responsiveness.

1.2.5 Scalability

- Requests are distributed across multiple servers to prevent bottlenecks and ensure responsiveness.
- Similar article searches run in parallel to reduce processing time and improve efficiency.
- The system leverages cloud computing resources to scale dynamically as the number of users and processed articles increases.
- Asynchronous processing ensures that time-intensive analyses do not hinder real-time user interactions.

1.2.6 Marketability

- The system is designed to appeal to a broad spectrum of users, including casual readers, journalists, and researchers, showcasing clear benefits such as quick fact-checking and source reliability.
- The system's brand and messaging is crafted to highlight its role in combating misinformation, increasing public trust and adoption.
- The system offers flexible integration options freemium, premium and enterprise options to facilitate partnerships with news outlets, social media platforms, and research organizations.
- The system utilizes a highly reliable dataset comprising credible news sources, enabling it to compare the source of news articles against the reference database.
- The system has a mobile application compatible with both Android and iOS environments designed to provide a user-friendly and seamless experience on mobile devices.

1.2.7 Extendibility

- The system's architecture is modular, allowing new features—such as additional language support or advanced NLP models—to be integrated with minimal disruption.

- The codebase and database structures are organized in a manner that supports easy inclusion of updated or alternative fact-checking sources and models.

1.2.8 Security

- The system ensures that all collected user data, including user information, visited articles, and feedback, is handled in strict compliance with relevant data protection regulations (GDPR).
- The system employs end-to-end encryption for data transmission and securely stores user passwords using Werkzeug Security,
- The system will implement robust authentication mechanisms.

1.2.9 Maintainability

- The system's source code follows standardized coding conventions and best practices, making it easier for new developers to understand and modify.
- Comprehensive documentation will be provided, covering architecture, APIs, and deployment procedures.

1.2.10 Flexibility

- The system is implemented on all of the popular browsers (Google Chrome, Opera, Microsoft Edge, Brave, Mozilla Firefox, Safari) and mobile platforms (iOS, Android).
- Mobile application ensures users can seamlessly access fact-checking features and reliability scores on the go.

1.2.11 Aesthetics

- The user interface has a clean, modern design that aligns with current web interface trends, ensuring visual appeal without distracting from core functionality.
- Consistent color themes, typography, and iconography will be used for readability and brand identity.
- Visual cues (alerts, hover-over and loading animations) enhance the user experience, guiding user interactions and clarifying complex information

1.3 Definitions, acronyms, and abbreviations

API: An API, or Application Programming Interface, is a collection of rules and protocols that allow software applications to interact and share data, features, or functionality. API

communication can be understood as an exchange of requests and responses between a client and a server. The client is the application sending the request, while the server processes the request and sends back a response. The API acts as the intermediary, facilitating this connection between the two [6].

NLP: Natural Language Processing (NLP) is a branch of computer science and artificial intelligence focused on enabling computers to comprehend human language. It combines computational linguistics, which examines the mechanics of language, with statistical methods, machine learning, and deep learning models. These techniques equip computers to analyze and interpret text or speech data, understanding their overall meaning, as well as the speaker's or writer's intentions and emotions [7].

ML: Machine learning (ML) is a branch of artificial intelligence (AI) and computer science that focuses on using data and algorithms to enable AI to imitate the way that humans learn, gradually improving its accuracy [8].

URL: The location of a webpage or file on the Internet [9].

GDPR: The General Data Protection Regulation, or GDPR, is a European Union (EU) law that governs how organizations within and outside the EU handle the personal data of EU residents. GDPR was adopted by the European Parliament and Council of the EU in 2016 and took effect on 25 May 2018 [10].

EFCSN: The European Fact-Checking Standards Network [11]

AI: Artificial intelligence (AI) refers to the capability of a digital computer or a robot controlled by computers to carry out tasks that are typically linked to intelligent entities. This term is often used to describe the endeavor of creating systems that possess human-like intellectual abilities, such as reasoning, uncovering meaning, generalizing, or learning from previous experiences. [12]

1.4 Overview

Checkmate is a tool designed to analyze the reliability, credibility, subjectivity and political bias of news articles by leveraging advanced NLP models and a robust backend architecture. Users can submit articles for evaluation, after which the application processes the content

through multiple analytical components. The language analyzer model assesses the grammar and linguistic structures of the article, the political bias classification model classifies the article along the political spectrum, and the image verification module performs reverse searches on images within the article to validate their context and authenticity. The backend aggregates these results to calculate a credibility score and generates a detailed report outlining the reasoning behind the evaluation. Additionally, users can compare articles with similar content and access previously analyzed reports for further reference.

The application also includes comprehensive account management and customization features. Users can create accounts, log in, and manage their profiles by changing passwords, updating payment plans, or upgrading to premium. For those using browser extensions, the application allows settings to be modified to enhance functionality. The application ensures data privacy and provides mechanisms to handle common issues, such as login or signup errors. Feedback from users is received and stored to improve the overall system performance.

With these capabilities, Checkmate provides an efficient and systematic approach to evaluating news articles. Its integration of AI-driven analysis ensures that the credibility and bias of content are assessed with precision, enabling users to critically engage with information. The combination of technical rigor and user-oriented design makes Checkmate a valuable resource for fostering media literacy and informed decision-making.

2. Current Software Architecture

2.1 Manual Competitors

- Teyit.org: A crowd-sourced fact-checking platform that relies on human expertise rather than automation [13].
- Meta (parent company of Facebook and Instagram) fact-checking: Employs human fact-checkers to review content flagged as potentially false. This approach struggles with scalability and is often too slow to address the rapid spread of misinformation across Meta's platforms [1].
- Reddit based fact-checking: Collaborative fact-checking efforts through user contributions and discussion on the Reddit platform.
- Community Notes on X (formerly Twitter): A crowdsourced approach that relies on users to provide contextual information and fact-checks for various contents [2].

These manual solutions provide sophisticated analysis but suffer from limited scalability and speed, especially during high-activity periods such as elections or globally sensitive events.

2.2 Automated Competitors

- InVID and WeVerify: Browser extensions specifically designed for video and image verification, but they lack text analysis capabilities [14].
- SurfSafe: Chrome extension focuses on identifying the source of images to determine legitimacy but doesn't analyze accompanying text content [15].
- Google Fact Check Tools: Web tool compiles and indexes fact-check reports from reputable sources but is reactive rather than proactive, as it only works for previously fact-checked claims [16].
- ClaimBuster and Hoaxy: Tools designed to identify claims that should be fact-checked or to track the flow of misinformation, primarily targeting researchers and organizations rather than general users [17, 18].

The key advantage of CheckMate over these solutions is its comprehensive approach that integrates multiple verification methods (text analysis, multimedia verification, political bias detection, objectivity analysis and source analysis) into a single, user-friendly browser extension and mobile application designed for real-time analysis and immediate output to users.

2.3 Limitations of Current Solutions

Current solutions in the fact-checking ecosystem exhibit several key limitations:

- Reactive vs. Proactive: Most existing solutions are reactive, addressing content only after it has been flagged or has already spread.
- Specialization vs. Integration: Current automated tools typically focus on specific aspects (only text or only multimedia) without offering holistic comprehensive analysis.
- Scalability Challenges: Manual fact-checking cannot keep pace with the volume of content being produced daily.
- Bias Detection Gap: Few solutions such as GroundNews [19] comprehensively address political bias detection alongside factual accuracy.
- Accessibility: Many sophisticated tools are designed for professional fact-checkers or researchers rather than everyday internet users [17, 18].

CheckMate aims to address these limitations through an integrated NLP-driven approach that provides immediate, comprehensive analysis of both factual accuracy and political bias, making fact-checking accessible to everyday users through a convenient browser extension. CheckMate addresses these limitations in existing tools by offering:

- Real-time analysis instead of just reactive verification
- Integration of both textual and multimedia analysis
- Political bias detection alongside factual verification
- User-friendly interface for the general public
- Automated processing that scales better than manual solutions

3. Proposed Software Architecture

3.1 Overview

The system is designed as a browser extension compatible with Chromium-based browsers such as Google Chrome, Opera, Microsoft Edge, Brave [5], and other common browsers such as Mozilla Firefox and Safari. Additionally, it is also available as mobile apps for both iOS and Android environments.

CheckMate comprises several external services, core analyzers, databases and score calculation working together to give reliability scores to news.

3.1.1. External Services

- Amazon RDS handles the management of our database, which includes three tables: user information, news reliability results, and news source reference tables [20].
- Amazon EC2 servers provide the necessary endpoints for our backend and host NLP models which are similarity, political bias classification, and objectivity classification models [21].
- Google Custom Search API handles text-based reverse searches to validate article content [22].
- Google Cloud Vision API handles image analysis and reverse searches [23].

3.1.2. Core Analyzers

The CheckMate architecture consists of 6 main components that work together to provide fact-checking and political bias detection:

- Reliability Analyzer Component: ML model that analyzes the reliability of the news using the similarity and credibility scores, political and objectivity classification results with image analysis results.
- Reverse Search Component: Reverse searches the title of the news using Google Custom Search API to find relevant news.
- Similarity Analyzer Component: Semantic similarity model that analyzes the similarity of the news article with the articles that were found through reverse search using Google Custom Search API.
- Content Analyzer Component: Analyzes the textual content of news articles using political bias classification and objectivity classification models to determine political inclination and bias.
- Source Analyzer Component: Cross references the source of the news with our reference database [24] to establish the credibility score of the news source.
- Multimedia Analyzer Component: Analyzes images on the news using reverse search to verify if they've been used out of context or manipulated using Google Cloud Vision API.

3.1.3. Data & Databases

Amazon RDS handles the management of our database, which includes five tables, through PostgreSQL.

- News Source Reference table stores the known trusted news sources with their credibility labels gathered from [24].
- All Articles table holds the outcomes of each analysis with reliability scores, credibility scores, political bias classification results, similarity results, objectivity classification results and image analysis results with news URL and news title. It also holds the last search date for the news' reliability score to get updated every week.
- User Information table holds the details of registered users, including attributes such as user ID, email address, password (hashed for security), creation date, subscription plan and Google ID (for sign in with Google option). This table is essential for managing user authentication, profile information, and activity tracking within the system.
- Similar Articles table stores the 10 reverse-searched articles, including each article's title, URL, similarity score (from the similarity analyzer component), and search date. Reverse search is performed using the searched news text via the Google Custom Search API.

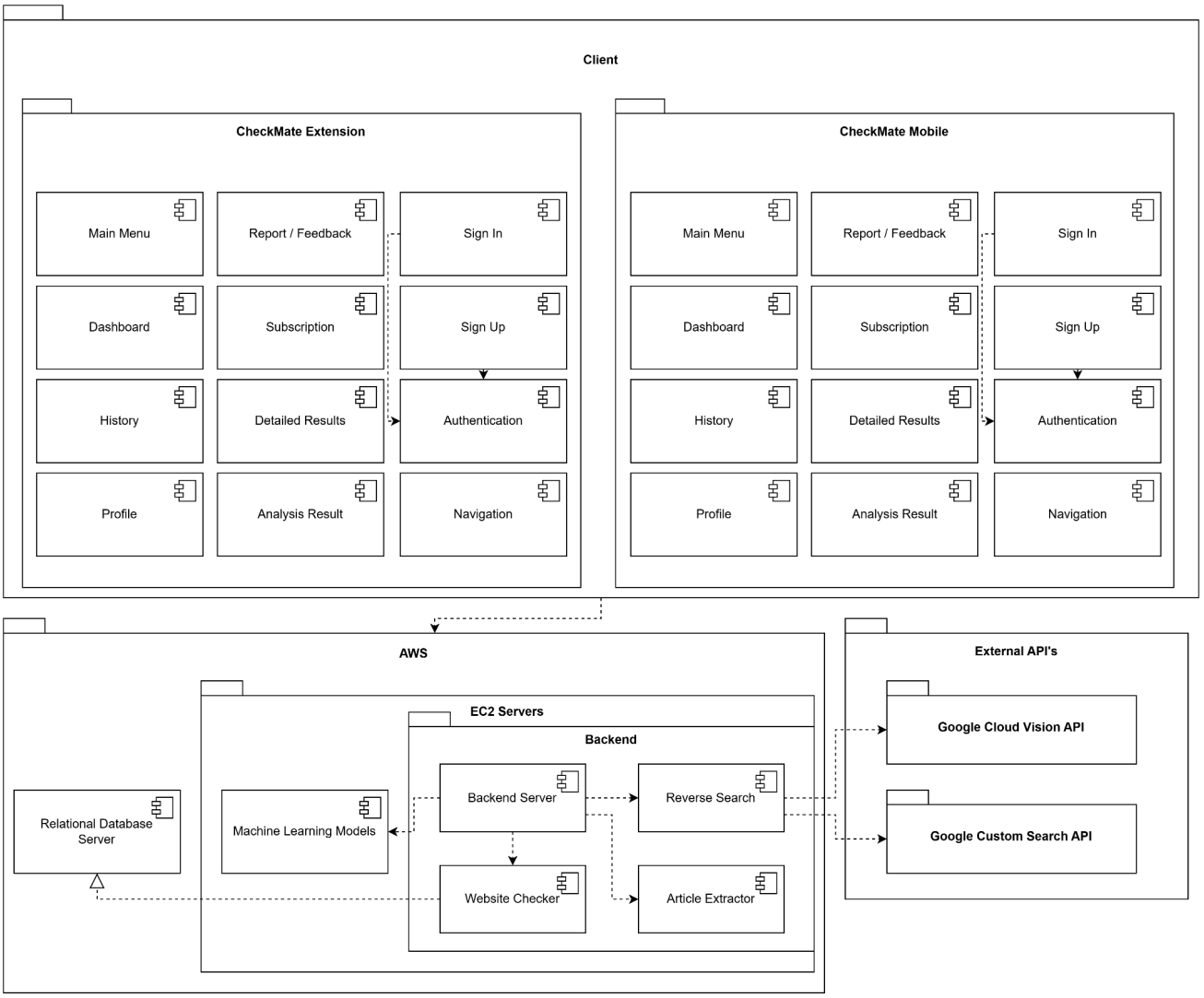
- Articles Searched by User table holds the news searched by the user with search date.

3.1.4. Score Calculation

- Reliability Score Calculator model uses the similarity and credibility scores, political bias and objectivity classification results with image analysis results to give a reliability score to the news.
- Similarity Score Calculator model compares the news' text with other articles found by searching the original news' title via Google Custom Search API.
- Credibility Score calculation is done by referencing the news' source within our News Source Reference table.

By orchestrating these components through well-defined interfaces, CheckMate efficiently processes news from multiple angles content, source, and multimedia to deliver a comprehensive reliability and credibility assessment, which is then stored for future reference and user queries.

3.2 Subsystem Decomposition



CheckMate consists of four main layers: Client Layer, AWS Layer, External APIs Layer, and Backend Logic Layer.

3.2.1 Client Layer

The client layer includes the user interfaces for CheckMate Extension and CheckMate Mobile, allowing users to interact with the system. Each client interface provides access to key functionalities such as authentication (sign-in and sign-up), dashboard, history, profile management, detailed results, and analysis reports. The client layer interacts with the backend services via authentication and navigation modules.

3.2.2 Backend Logic Layer (AWS)

The backend logic is hosted on AWS EC2 servers and handles the system's core functionalities. This layer consists of:

Backend Server: Manages communication between the client layer and the processing components.
Website Checker: Evaluates the credibility of a given website.
Reverse Search: Performs similar article extraction and fact-checking using external APIs.
Article Extractor: Extracts and analyzes textual content from web sources.
Machine Learning Models: Provides AI-driven analysis for misinformation detection.
The backend layer interacts with a relational database to store and retrieve necessary data, ensuring efficient system operations.

3.2.3 External APIs Layer

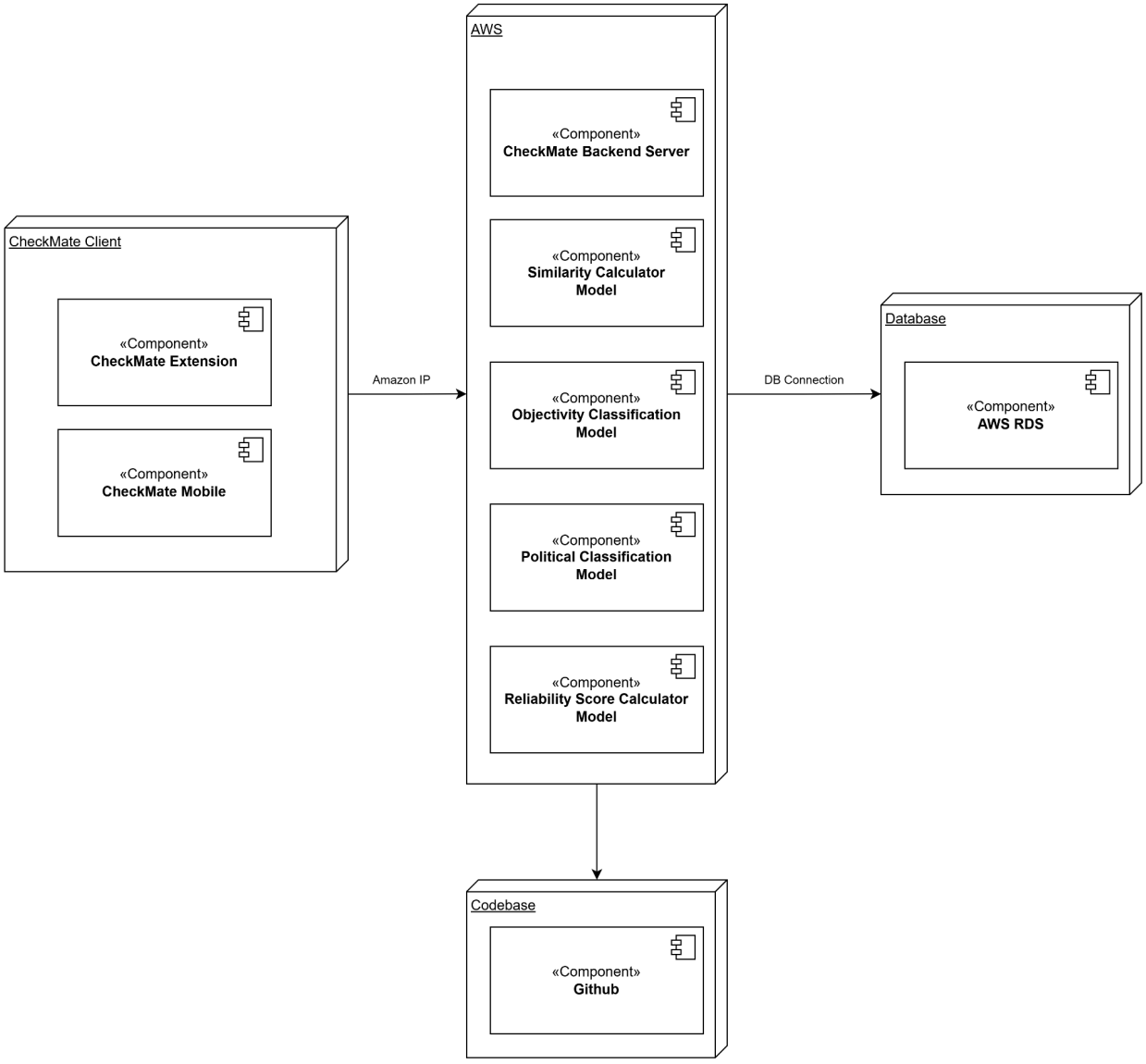
The External APIs layer enhances CheckMate's capabilities by integrating third-party services:

Google Custom Search API: Retrieves web results related to fact-checking queries [22].
Google Cloud Vision API: Facilitates image recognition and analysis for reverse searching [23].

3.2.4 Interactions Between Layers

Users interact with the Client Layer through the extension or mobile application.
Requests are processed by the Backend Logic Layer, which communicates with AWS services and machine learning models.
If external data is needed, the backend interacts with Google Cloud Vision and Custom Search APIs for image and web search results.
The relational database stores relevant data for future analysis and system optimization.
This layered architecture ensures modularity, scalability, and efficient data flow, enabling CheckMate to provide accurate and real-time fact-checking results.

3.3 Hardware/software mapping



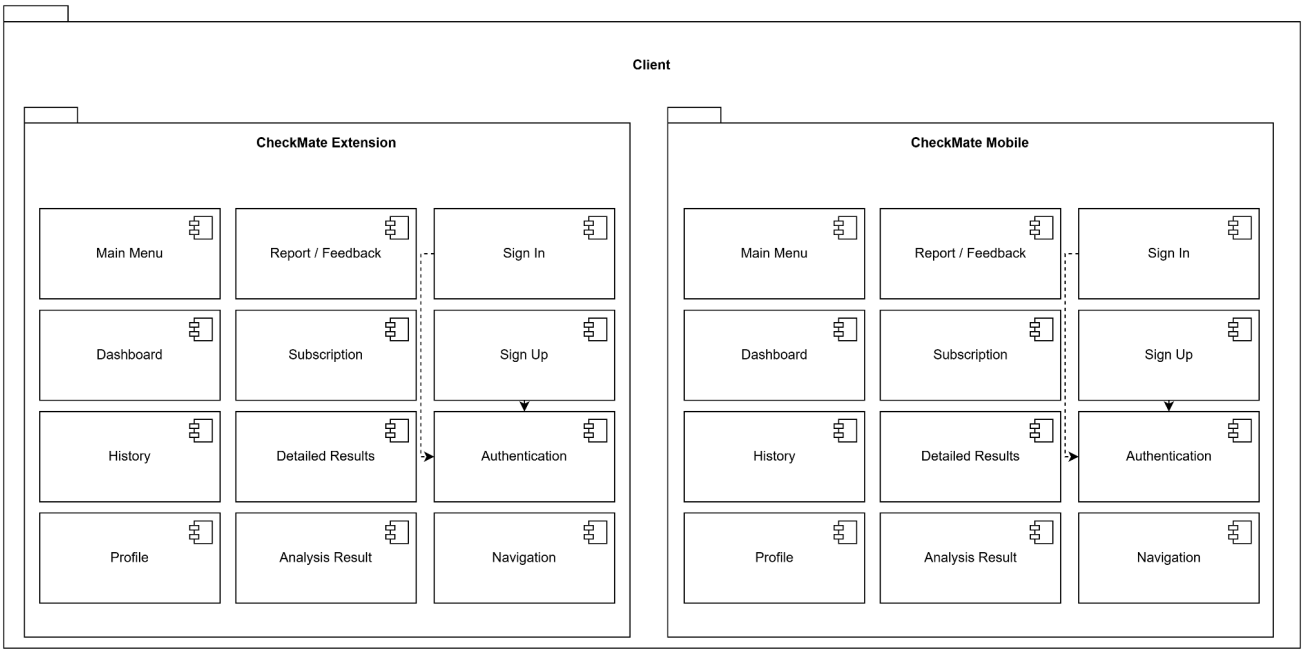
The CheckMate system consists of four main modules in its hardware/software mapping: CheckMate Client, AWS, Database, and Codebase. The CheckMate Client module includes the CheckMate Extension and CheckMate Mobile, which provide the user interface for interacting with the system. These clients communicate with the CheckMate Backend Server, which is deployed on AWS. The backend is responsible for handling requests and running machine learning models, including the Similarity Calculator Model, Objectivity Classification Model, Political Bias Classification Model, and Reliability Score Calculator Model, to analyze and classify input data. The AWS RDS module serves as the database, storing user interactions, analysis results, and system data. Additionally, the Codebase module is hosted on GitHub, where the source code is maintained and updated. This structure ensures that CheckMate is scalable, secure, and capable of handling multiple user requests efficiently.

3.4 Access control and security

CheckMate implements a tiered access control system to manage user permissions and security across its Free, Premium, and Enterprise models. Authentication and authorization are handled through a secure backend, ensuring that users can only access features allowed by their subscription tier. Free users have limited daily usage, including a cap on fact-checking requests and analysis reports. Premium users gain extended access with higher daily limits and extra outputs from AI models. Enterprise users receive unrestricted access, custom API access. All user data is securely stored in AWS RDS, with strict access controls to prevent unauthorized modifications. Additionally, role-based authentication ensures that sensitive operations, such as modifying subscription plans or accessing administrative tools, are restricted to authorized users only.

4. Subsystem Services

4.1 Client Subsystem

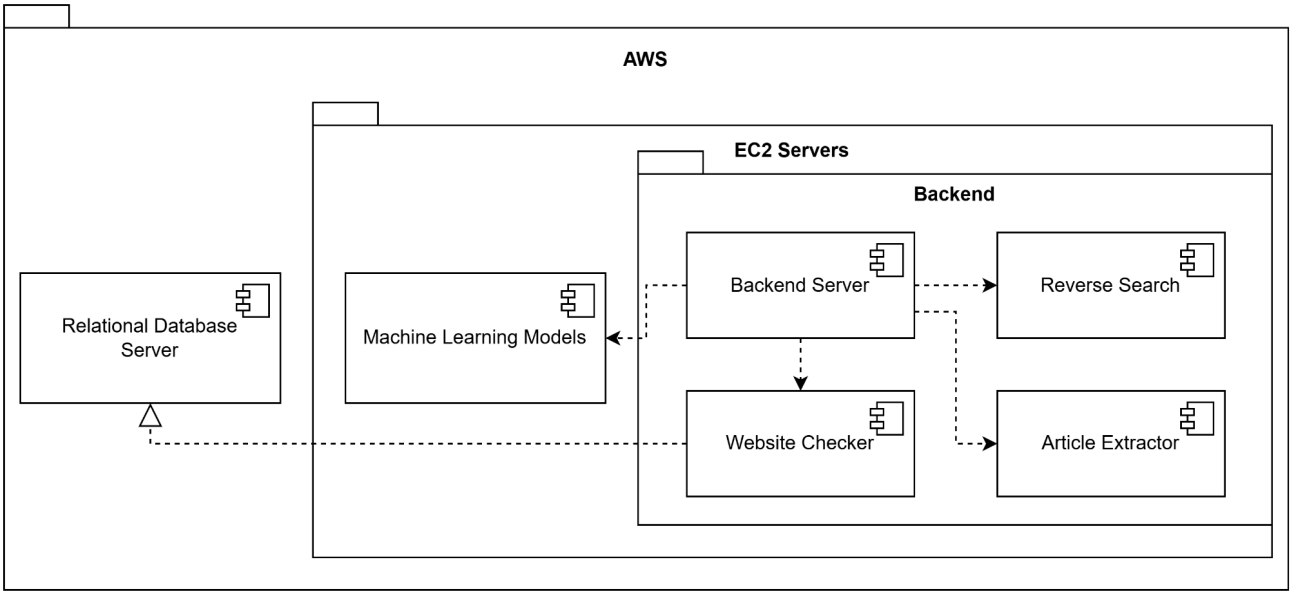


The Client Subsystem consists of the CheckMate Extension and CheckMate Mobile, which serve as the primary interfaces for users. These components allow users to submit URLs, articles, or text for fact-checking, view analysis results, and interact with various features. The client communicates with the backend through API requests, ensuring seamless data exchange between the frontend and the backend.

4.1.1 Components of the Client Subsystem

- Main Menu: Serves as the entry point for users, providing access to all primary functionalities such as fact-checking, history, and profile settings.
- Dashboard: Displays summarized analysis results, notifications, and system recommendations based on recent activities.
- History: Stores previously analyzed articles, allowing users to revisit past fact-checking results.
- Profile: Manages user information, including subscription status (Free, Premium, Enterprise) and personalized settings.
- Report/Feedback: Allows users to submit feedback on fact-checking results, suggest new sources, or report inaccuracies.
- Subscription: Handles user access levels, showing available plans and processing upgrades to Premium or Enterprise models.
- Detailed Results: Provides in-depth analysis, including similarity scores, objectivity ratings, political bias classification, and reliability scores.
- Analysis Result: A summary of the overall fact-checking findings, displayed in a user-friendly format.
- Sign In / Sign Up: Manages user authentication and account creation.
- Authentication: Ensures secure user login and verifies subscription status for access to premium features.
- Navigation: Handles internal transitions between different sections of the client application.

4.2 AWS Subsystem



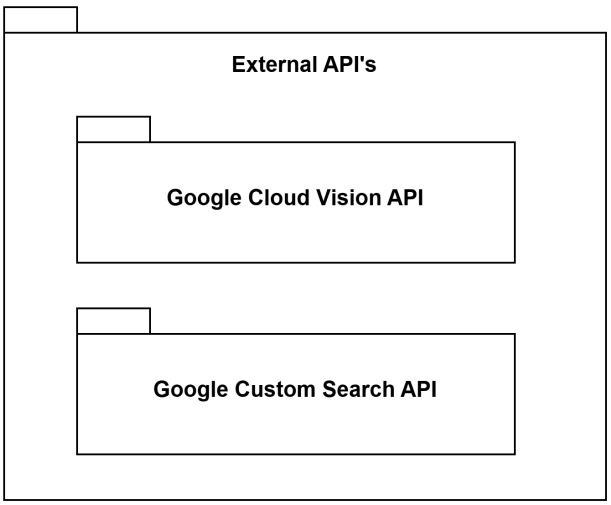
The AWS Subsystem forms the core backend infrastructure, responsible for handling fact-checking requests, processing machine learning models, managing data, and ensuring system reliability. The backend is designed to efficiently process high volumes of requests while maintaining security and scalability.

4.2.1 Components of the AWS Subsystem

- CheckMate Backend Server: The central processing unit of the system that manages user requests, processes authentication, retrieves data from the database, and coordinates interactions between machine learning models and external APIs.
- Website Checker: A module within the backend that verifies URLs submitted by users, extracting relevant content for fact-checking.
- Article Extractor: Processes raw text from submitted URLs or user input, preparing it for analysis by different machine learning models.
- Reverse Search: Searches the web for similar articles, helping determine if a claim has already been fact-checked by trusted sources.
- Machine Learning Models: A set of AI-driven components that analyze the extracted text and generate credibility scores:
    - Similarity Calculator Model: Uses natural language processing (NLP) to compare the submitted claim with existing fact-checked content, determining how closely it matches known misinformation.
    - Objectivity Classification Model: Evaluates whether the text is neutral, opinionated, or highly biased, helping users identify potential misinformation.

    - Political Bias Classification Model: Assesses the political alignment of the text, providing insights into potential ideological biases.
    - Reliability Score Calculator Model: Aggregates multiple factors such as source credibility, historical accuracy, and fact-checking results to generate a final reliability score.
- Relational Database Server (AWS RDS): Stores all user data, analysis results, fact-checking history, and system logs in a structured format. The database ensures secure and efficient data retrieval.

4.3 External APIs Subsystem



The External APIs Subsystem enhances CheckMate's capabilities by integrating third-party services for additional data retrieval, text processing, and image analysis.

4.3.1 Components of the External APIs Subsystem

- Google Cloud Vision API: Used for analyzing images, allowing CheckMate to extract information from images of news articles, social media posts. This ensures that misinformation shared through images can also be fact-checked.
- Google Custom Search API: Enables advanced reverse searching of web content to find original sources of claims. This helps determine whether an article or statement has been republished, manipulated, or taken out of context.

5. Test Cases

Test ID	TC_1.1 – Add Extension	Category	Functional	Severity	Critical
Objective	Verify the extension can be installed successfully.				
Steps	User installs the Checkmate extension.				
Expected	Extension installation succeeds with no errors (the extension icon is visible).				
Result	Pass				
Date	01.03.2025				

Test ID	TC_1.2 – Sign Up with Valid Data	Category	Functional	Severity	Critical
Objective	Verify that a new account is created with valid data.				
Steps	User clicks Sign Up. User enters valid name, email, phone number, password, confirm password. User clicks Sign Up.				
Expected	The user is added to the database otherwise an appropriate error is given.				
Result	Pass				
Date	01.03.2025				

Test ID	TC_1.3 – Sign Up with Missing Fields	Category	Functional	Severity	Major
Objective	Ensure an error is shown if any required fields are omitted.				
Steps	User clicks Sign Up. User leaves at least one required field blank (e.g., phone or password). User clicks Sign Up.				
Expected	Error message appears prompting the user to fill the missing field(s).				

Result	Pass
Date	01.03.2025

Test ID	TC_1.4 – Sign Up with Invalid Email	Category	Functional	Severity	Major
Objective	Confirm that an invalid email format is not accepted.				
Steps	User clicks Sign Up. User enters an invalid email (e.g., “test@” or missing “@”). User fills other fields correctly. User clicks Sign Up.				
Expected	An error message is displayed (e.g., “Invalid email format”); account is not created.				
Result	Fail				
Date	01.03.2025				

Test ID	TC_1.5 – Sign Up with Password Mismatch	Category	Functional	Severity	Minor
Objective	Ensure password and confirm password must match.				
Steps	User clicks Sign Up. User enters a valid password in “Password” but a different value in “Confirm Password.” User clicks Sign Up.				
Expected	Error message indicates the passwords do not match; account is not created.				
Result	Pass				
Date	01.03.2025				

Test ID	TC_1.6 – Sign Up with Existing Email	Category	Functional	Severity	Major
Objective	Check that the system prevents duplicate accounts with the same email.				
Steps	User clicks Sign Up. User enters an email that already exists in the system. User fills in other fields and clicks Sign Up.				
Expected	Error message “Email already exists” (or similar); account creation is blocked.				
Result	Pass				
Date	01.03.2025				

Test ID	TC_2.1 – Log In with Valid Credentials	Category	Functional	Severity	Major
Objective	Ensure that users can log in with the correct email/password.				
Steps	User opens the extension. User clicks Sign In. User enters a valid email and password. User clicks Sign In.				
Expected	User is redirected to the MainMainPage				
Result	Pass				
Date	01.03.2025				

Test ID	TC_2.2 – Log In with Invalid Password	Category	Functional	Severity	Major
Objective	Verify incorrect passwords are rejected.				
Steps	User opens the extension. User clicks Sign In.				

	User enters the correct email but an incorrect password. User clicks Sign In.
Expected	An error message is displayed (e.g., “Invalid password”); login fails.
Result	Pass
Date	01.03.2025

Test ID	TC_2.3 – Log In with Non-Existent Email	Category	Functional	Severity	Major
Objective	Check that users cannot log in using an unregistered email.				
Steps	User opens the extension. User clicks Sign In. User enters an email not found in the database. User clicks Sign In.				
Expected	Error message “Email not found” (or similar) is displayed.				
Result	Pass				
Date	01.03.2025				

Test ID	TC_2.4 – Log In with Empty Fields	Category	Functional	Severity	Major
Objective	Ensure the system requires both email and password.				
Steps	User opens the extension, clicks Sign In. User leaves email or password blank. User clicks Sign In.				
Expected	System displays an error indicating missing credentials.				
Result	Pass				
Date	01.03.2025				

Test ID	TC_3.1 – Standard Logout	Category	Functional	Severity	Major
Objective	Ensure that clicking logout signs the user out.				
Steps	User clicks the logout icon.				
Expected	User is signed out and redirected to the login page (or sees a logged-out state).				
Result	Pass				
Date	01.03.2025				

Test ID	TC_3.2 – Logout via Inactivity	Category	Functional	Severity	Minor
Objective	Check if the system auto-logs out a user after inactivity.				
Steps	User logs in, then remains idle for the configured time limit.				
Expected	System logs the user out automatically, requiring re-login.				
Result	Pass				
Date	01.03.2025				

Test ID	TC_4.1 – Analyze a Valid Article Page	Category	Functional	Severity	Critical
Objective	Verify that analyzing a valid news/article page works.				
Steps	User clicks Analyze Current Page.				
Expected	System displays reliability score, analysis summary, similar articles, and any image analysis.				
Result	Fail. Image analysis is missing				
Date	01.03.2025				

Test ID	TC_4.2 – Analyze a Non-Article Page or Invalid Content	Category	Functional	Severity	Major
Objective	Ensure the system handles invalid pages				
Steps	User is on a page that is not a typical article (e.g., a blank page). User clicks Analyze Current Page.				
Expected	System either shows an error (“Unable to analyze”) or minimal results.				
Result	Fail				
Date	01.03.2025				

Test ID	TC_5.1 – Analyze Valid URL	Category	Functional	Severity	Critical
Objective	Verify user can input a URL to be analyzed with the same reliability features.				
Steps	User enters a valid URL in the input box. User clicks Analyze URL.				
Expected	System displays reliability score, similar articles, image analysis, etc.				
Result	Fail. Image analysis is missing.				
Date	01.03.2025				

Test ID	TC_5.2 – Analyze Invalid or Empty URL	Category	Functional	Severity	Major
Objective	Check system response to an invalid or no URL.				
Steps	User leaves the URL box empty or types a malformed URL (e.g., http://notvalid). User clicks Analyze URL.				
Expected	Error message prompts user that the URL is invalid, or to fill in the input box.				
Result	Fail. Invalid URL can still be entered.				

Date	01.03.2025

Test ID	TC_6 – View More Details for Valid Analysis	Category	Functional	Severity	Major
Objective	Verify that clicking “Show more details” loads additional information about the article.				
Steps	After the article is analyzed, user clicks Show More Details.				
Expected	System shows extra details (stats, references, etc.).				
Result	Pass				
Date	01.03.2025				

Test ID	TC_7.1 – Change Plan to Premium	Category	Functional	Severity	Major
Objective	Ensure a user can upgrade from free to premium.				
Steps	User clicks profile icon. User selects “Premium” plan.				
Expected	Plan changes to premium; user sees updated subscription status.				
Result	Pass				
Date	01.03.2025				

Test ID	TC_7.2 – Change Plan to Enterprise	Category	Functional	Severity	Major
Objective	Verify a user can change from premium/free to enterprise.				
Steps	User clicks profile icon. User selects “Enterprise” plan.				
Expected	User subscription changes to enterprise plan.				

Result	Pass
Date	01.03.2025

Test ID	TC_7.3 – Change Plan to Free	Category	Functional	Severity	Major
Objective	Check a user can revert back to free.				
Steps	User clicks profile icon. Selects “Free” plan.				
Expected	Subscription changes to free; possibly showing a downgrade confirmation.				
Result	Pass				
Date	01.03.2025				

Test ID	TC_7.4 – Change Plan with Payment Failure	Category	Functional	Severity	Major
Objective	See what happens if payment for a paid plan fails.				
Steps	User selects a paid plan. Payment gateway returns a failure or declines the card.				
Expected	System notifies user of payment failure; plan remains unchanged.				
Result	Fail				
Date	01.03.2025				

Test ID	TC_8.1 – Change Password Successfully	Category	Functional	Severity	Minor
Objective	User can change their current password to a new one.				
Steps	User clicks profile icon. User selects Change Password.				

	User enters the new password and confirms it correctly. User clicks Change Password button.
Expected	Password updates; user sees success message.
Result	Pass
Date	01.03.2025

Test ID	TC_8.2 – Change Password with Mismatched Confirmation	Category	Functional	Severity	Minor
Objective	Ensure mismatched new passwords are detected.				
Steps	User enters new password but a different confirm password. User clicks Change Password.				
Expected	Error message: “Passwords do not match.”				
Result	Pass				
Date	01.03.2025				

Test ID	TC_8.3 – Change Password with Blank Fields	Category	Functional	Severity	Minor
Objective	Check that all fields are required.				
Steps	User leaves the new password or confirm password field empty. User clicks Change Password.				
Expected	Error message: “Please fill all required fields.”				
Result	Pass				
Date	01.03.2025				

Test ID	TC_9 – View Dashboard	Category	Functional	Severity	Major

Objective	Check that clicking “Your Dashboard” shows correct stats.
Steps	User logs in. User clicks MainMenuPage → “Your Dashboard” (or Profile → Dashboard).
Expected	Dashboard displays daily limit, # of articles analyzed, average reliability score, total articles.
Result	Pass
Date	01.03.2025

Test ID	TC_10 – View History	Category	Functional	Severity	Major
Objective	Check that clicking “Your Dashboard” shows correct stats.				
Steps	User logs in. User clicks MainMenuPage → “Your Dashboard” (or Profile → Dashboard).				
Expected	Dashboard displays daily limit, # of articles analyzed, average reliability score, total articles.				
Result	Pass				
Date	01.03.2025				

Test ID	TC_11 – Send Report	Category	Functional	Severity	Major
Objective	Confirm a user can file a “mistake” report.				
Steps	User clicks Report Icon or Report Mistake. User selects “Mistake” as report type. User enters a message. User clicks Send Report.				
Expected	Report is sent to admins.				
Result	Pass				
Date	01.03.2025				

Test ID	TC_12.1 – Forgot Password with Valid Email	Category	Functional	Severity	Major
Objective	Ensure the user can reset their password using a correct email and code.				
Steps	User clicks Sign In → Forgot Password link. User enters their valid email and clicks Send Verification Code. User receives the code by email, enters it, and sets a new password. User confirms new password.				
Expected	Password is updated; user can log in with the new password.				
Result	Pass				
Date	01.03.2025				

Test ID	TC_12.2 – Forgot Password with Unregistered Email	Category	Functional	Severity	Minor
Objective	Verify the system rejects invalid/unregistered emails for password reset.				
Steps	User clicks Forgot Password. User enters an email not in the system. User clicks Send Verification Code.				
Expected	Error message: “Email not found” or similar.				
Result	Fail				
Date	01.03.2025				

Test ID	TC_12.3 – Forgot Password with Invalid Verification Code	Category	Functional	Severity	Major
Objective	Check that an incorrect code is not accepted.				
Steps	User enters a verification code that is wrong or expired. User tries to set a new password.				
Expected	Error message indicates invalid or expired code.				

Result	Fail. Code doesn't expire.
Date	01.03.2025

Test ID	TC_12.4 – Forgot Password with Mismatched New Passwords	Category	Functional	Severity	Major
Objective	Ensure new password and confirm password must match.				
Steps	User enters a valid verification code. User types a new password but a different confirm password. User clicks Submit.				
Expected	Error message “Passwords do not match.”				
Result	Pass				
Date	01.03.2025				

Test ID	TC_13 – View Credibility after Analyzing	Category	Functional	Severity	Critical
Objective	User can see if a website is “credible,” “mixed,” or “uncredible.”				
Steps	User clicks Analyze Current Page or inputs URL. After results load, user clicks More Details.				
Expected	System displays credibility status (“credible,” “mixed,” or “uncredible”).				
Result	Pass				
Date	01.03.2025				

Test ID	TC_14.1 – Similarity Detection Model	Category	Functional	Severity	Critical

Objective	Verify the similarity detection model correctly identifies text similarity.
Steps	Provide two identical text inputs to the model.
Expected	Output = 1 (identical texts).
Result	Pass
Date	01.03.2025

Test ID	TC_14.2 – Similarity Detection Model	Category	Functional	Severity	Critical
Objective	Verify the similarity detection model correctly identifies text similarity.				
Steps	Provide two completely different with no semantic similarity text inputs to the model.				
Expected	Output = 0 (no similarity).				
Result	Pass				
Date	01.03.2025				

Test ID	TC_14.3 – Similarity Detection Model	Category	Functional	Severity	Critical
Objective	Verify the similarity detection model correctly identifies text similarity.				
Steps	Provide two partially similar text inputs to the model.				
Expected	Output = a value between 0 and 1 (partial similarity).				
Result	Pass				
Date	01.03.2025				

Test ID	TC_15.1 – Objectivity Classification Model	Category	Functional	Severity	Critical
Objective	Verify the objectivity classification model correctly classifies text as objective or subjective.				
Steps	Provide a factual news article (e.g., "The population of New York is 8.5 million")..				
Expected	Output = "Objective".				
Result	Pass				
Date	01.03.2025				

Test ID	TC_15.2 – Objectivity Classification Model	Category	Functional	Severity	Critical
Objective	Verify the objectivity classification model correctly classifies text as objective or subjective.				
Steps	Provide an opinion piece (e.g., "New York is the best city in the world").				
Expected	Output = "Subjective".				
Result	Pass				
Date	01.03.2025				

Test ID	TC_16.1 – Political Bias Classification Model	Category	Functional	Severity	Critical
Objective	Verify the political bias classification model correctly identifies bias as left, center, or right.				
Steps	Provide a text with left-leaning political bias (e.g., "The government should increase taxes on the wealthy to fund social programs").				

Expected	Output = "Left".
Result	Pass
Date	01.03.2025

Test ID	TC_16.2 – Political Bias Classification Model	Category	Functional	Severity	Critical
Objective	Verify the political bias classification model correctly identifies bias as left, center, or right.				
Steps	Provide a text with right-leaning political bias (e.g., "Lower taxes for businesses will stimulate economic growth").				
Expected	Output = "Right".				
Result	Pass				
Date	01.03.2025				

Test ID	TC_16.3 – Political Bias Classification Model	Category	Functional	Severity	Critical
Objective	Verify the political bias classification model correctly identifies bias as left, center, or right.				
Steps	Provide a politically neutral text				
Expected	Output = "Center".				
Result	Pass				
Date	01.03.2025				

Test ID	TC_17.1 – Reliability Score Model	Category	Functional	Severity	Critical
Objective	Verify the reliability scoring model correctly assigns a reliability score between 0 and 1.				
Steps	Input a text article to the model and check if the model outputs a reliability score.				
Expected	A reliability score between 0-1 should be retrieved.				
Result	Fail				
Date	01.03.2025				

Test ID	TC_17.2 – Reliability Score Model	Category	Functional	Severity	Critical
Objective	Verify the reliability scoring model correctly assigns a reliability score between 0 and 1.				
Steps	Input a fake news article and receive an output				
Expected	A reliability score between 0-0.5 should be retrieved.				
Result	Fail				
Date	01.03.2025				

Test ID	TC_17.3 – Reliability Score Model	Category	Functional	Severity	Critical
Objective	Verify the reliability scoring model correctly assigns a reliability score between 0 and 1.				
Steps	Input a real news article and receive an output				
Expected	A reliability score between 0.5-1 should be retrieved.				
Result	Fail				

Date	01.03.2025

Test ID	TC_18.1 – Edge Case Testing for Similarity Detection Model	Category	Edge …	Severity	Major
Objective	Verify the similarity detection model handles edge cases correctly.				
Steps	Provide two texts with only stopwords (e.g., "the and is").				
Expected	Output = 1 (stopwords are considered identical)				
Result	Pass				
Date	01.03.2025				

Test ID	TC_18.2 – Edge Case Testing for Similarity Detection Model	Category	Edge …	Severity	Major
Objective	Verify the similarity detection model handles edge cases correctly.				
Steps	Provide one text and an empty string as the second input.				
Expected	Output = 0 (no similarity with an empty string)				
Result	Pass				
Date	01.03.2025				

Test ID	TC_19 – Edge Case Testing for Objectivity Classification Model	Category	Edge …	Severity	Major
Objective	Verify the objectivity classification model handles ambiguous or mixed texts.				
Steps	Provide a text that mixes facts and opinions (e.g., "The population of New York is 8.5 million, and it's the best city in the world").				

Expected	Output = "Subjective" (since the text contains opinions).
Result	Pass
Date	01.03.2025

Test ID	TC_20 – Edge Case Testing for Political Bias Classification Model	Category	Edge …	Severity	Major
Objective	Verify the political bias classification model handles neutral or ambiguous texts.				
Steps	Provide a text with no political context (e.g., "The weather today is sunny").				
Expected	Output = "Center" (neutral).				
Result	Pass				
Date	01.03.2025				

Test ID	TC_21 – Integration of All Models	Category	Integra…	Severity	Critical
Objective	Verify all models work together seamlessly to provide a comprehensive fact-checking output.				
Steps	Provide a news article as input. Run the article through all models. • Similarity detection • Objectivity classification • Political bias classification • Reliability scoring				
Expected	Similarity detection output = value between 0 and 1. Objectivity classification output = "Objective" or "Subjective". Political bias classification output = "Left", "Center", or "Right". Reliability scoring output = value between 0 and 1.				
Result	Fail				
Date	01.03.2025				

Test ID	TC_22 – Claim Extraction	Category	Functional	Severity	Critical
Objective	Ensure that claim extraction is complete				
Steps	Input a news article text.				
Expected	At least one claim must be extracted.				
Result	Pass				
Date	01.03.2025				

Test ID	NFR-1 – Clear Visual Indicators	Category	Non-fu…	Severity	Major
Objective	Ensure color-coded scoring and political compass graphics are visible and understandable.				
Steps	User opens an article analysis page in the extension. Inspect that the reliability score is color-coded (e.g., green=credible, yellow=mixed, red=uncredible). Check for a political compass graphic or textual indicator.				
Expected	Colors, icons, or textual labels are clear and consistent				
Result	Pass				
Date	01.03.2025				

Test ID	NFR-2 – Browser Compatibility	Category	Compat…	Severity	Major
Objective	The system is accessible on all major Chromium browsers (Google Chrome, Opera, Microsoft Edge and Brave) as well as Mozilla Firefox and Safari.				
Steps	Use the extension				
Expected	No broken layouts or overlapping elements. All critical functionality remains accessible at different browsers.				

Result	Fail
Date	01.03.2025

Test ID	NFR-3 – Mobile UI Responsiveness & Usability	Category	Usability	Severity	Major
Objective	Validate that the mobile app's interface is optimized for smaller screens and different orientations (portrait/landscape).				
Steps	Open the app on various device sizes (phone and tablet). Navigate through menus, open analysis results, and check any interactive elements (e.g., text fields, buttons). (Optional) Rotate the device from portrait to landscape.				
Expected	No UI elements are cut off or overlapping. Buttons/fields are appropriately scaled and easy to tap. Navigation remains logical in both orientations.				
Result	Pass				
Date	01.03.2025				

Test ID	PERF-1 – Response Time for Browser Extension	Category	Perform…	Severity	Critical
Objective	Verify both browser extension and mobile app respond within acceptable time limits for all user actions.				
Steps	Perform article analysis request using the browser extension. Perform article analysis request using the mobile app.				
Expected	Response time ≤ 5 seconds for both browser extension and mobile app.				
Result	Pass				
Date	01.03.2025				

Test ID	M-1 – Complete Mobile App	Category	Functional	Severity	Critical
Objective	Ensure that core features (article analysis, reliability scoring, political compass, account management, etc.) are available and function correctly on the mobile app, matching the desktop extension.				
Steps	Launch the mobile app on a phone or tablet. Log in. Access and use each major feature (e.g., article analysis, viewing reliability scores, changing subscription plan).				
Expected	All primary features that exist on the browser extension are also present in the mobile app and function without major differences or missing steps.				
Result	Pass				
Date	01.03.2025				

Test ID	M-2 – Log in to the Mobile App	Category	Functional	Severity	Critical
Objective	Ensure that login works on the mobile application, and the token is generated.				
Steps	Open the app. Enter the email and the password. Press the login button.				
Expected	User is navigated to the homepage. Token is generated.				
Result	Pass				
Date	08.03.2025				

Test ID	M-3 – Log out of the Mobile App	Category	Functional	Severity	Critical
Objective	Ensure that the user logs out upon token expiry.				
Steps	Open the app. Log in.				

	Wait for 1 hour (token expiry).
Expected	User is automatically logged out.
Result	Pass
Date	08.03.2025

Test ID	M-5 – More Details page of Article Analysis in Mobile App	Category	Functional	Severity	Critical
Objective	Ensure that the user is automatically redirected to the article page after they analyze the article on the mobile app.				
Steps	Open the app. Login Press the ‘Search’ button that is on the bottom-right of the page. Enter the article url. Press the ‘Analyze’ button.				
Expected	User is navigated to the article page.				
Result	Pass				
Date	08.03.2025				

6. Consideration of Various Factors in Engineering Design

6.1 Constraints

Implementation Constraints

- The browser extension is compatible with all major Chromium browsers (Google Chrome, Opera, Microsoft Edge and Brave) as well as Mozilla Firefox and Safari [5].
- GitHub and Jira are used to track the deadlines, issues, and the codebase.
- HTML and CSS frameworks are used for frontend development.
- Python is used for machine learning (ML) development and backend development.
- PostgreSQL hosted by Amazon RDS server is used for the database system.
- Amazon Elastic Compute Cloud (Amazon EC2) server is used for ML model and backend deployment.

Economic Constraints

- Our project requires several external libraries, frameworks, and models. Therefore, our group has opted to use as many open-source frameworks as possible.
- Google Vision API [23], Google Custom Search API [22], ML model and backend deployment on Amazon EC2 servers use a paid plan after the free tier parameters are reached.

Ethical Constraints

- All interactions and data collected from the users are handled within data protection law General Data Protection Regulation (GDPR).
- The users are informed about the application's limitations and scope before registration.
- The system utilizes personal data and storing said data in the database hosted on Amazon RDS.
- No unnecessary user data will be collected.
- The system will clearly communicate to the user about its shortcomings.
- The system will clearly communicate reasons for reliability scores of the news articles and the reasons for credibility scores for news sources.
- The system will suggest trustworthy sources to the user about the news article that they are searching for.
- The system is designed to ensure objectivity and fairness, actively compensating for any biased outcomes to provide accurate labeling of news articles and news sources.

Language Constraints

- The system will work on news articles and new sources in English because of the lack of labeled Turkish news article datasets.

6.2 Standards

- The system will abide by the European Fact-Checking Standards Network (EFCSN) [11] as a benchmark in fact-checking news articles.
- The system will abide by Google Cloud Vision API Terms of Service, Bing Search API License Agreement, and other third-party API usage policies.
- The system will abide to General Data Protection Regulation (GDPR) [10]

7. Teamwork Details

7.1 Contributing and functioning effectively on the team

Our team collaborates using a combination of Agile-inspired methodologies, structured task management, and frequent communication channels to ensure that everyone contributes effectively to the project:

1. Project Management with JIRA
   - We use JIRA as our primary tool for planning and tracking tasks. Each sprint is broken down into subtasks, which are then assigned to individual team members.
   - JIRA's board view provides transparency into task status (e.g., To Do, In Progress, Done) and helps us monitor deadlines, prioritize workload, and spot any bottlenecks.

2. Regular Communication and Meetings
   - We maintain a WhatsApp group for rapid, day-to-day communication, allowing team members to quickly share updates, ask questions, and coordinate ad-hoc tasks.
   - In addition to ongoing online communication, we hold three weekly meetings to discuss progress, align on upcoming goals, and resolve any challenges. .

3. Role Division and Collaboration
   - Backend Team: Develops core server-side logic, database integrations, and APIs.
   - Frontend Team: Implements the user interface, browser extension features, and mobile app functionality.
   - Machine Learning Team: Focuses on model development, training, and integration with external services (e.g., NLP).
   - Despite these primary roles, we encourage knowledge sharing and cross-functional support. If someone encounters a roadblock, teammates from any domain can step in to offer insights.

4. Coordination and Accountability
   - Each member is responsible for updating JIRA tasks with progress notes and shifting status as they move from development to testing.
   - Even if there is a role division each member gets to work on a different team if another team needs their help.

By combining these practices, our team maintains clear communication, well-defined responsibilities, and a supportive environment that enables us to efficiently develop CheckMate's backend, frontend, and machine learning models.

7.2 Helping creating a collaborative and inclusive environment

Each team member takes the lead on a specific core component, but that doesn't prevent us from assisting one another with coding challenges. Whenever someone encounters an issue while coding we all work on it together by sharing the code with the issue among each other, ensuring that everyone is informed and can analyze the code to help find a solution.

Additionally, we use WhatsApp and Google meet to discuss problems and share our planned solutions, keeping the entire team updated. We recognize that this is a collaborative project, and although tasks are divided among us, we all share the responsibility of delivering a fully developed application.

7.3 Taking lead role and sharing leadership on the team

Our team embraces a domain-based leadership model, where individuals lead when they have expertise in a specific area. For example, someone well-versed in mobile app development guided that portion of the project, while another person with deep ML knowledge led model selection and training. However, all major decisions still go through a collaborative process, ensuring that everyone's perspectives are considered.

Domain leads also act as mentors, sharing knowledge and providing hands-on guidance to team members less familiar with their area of expertise. This approach fosters ownership, as each member takes initiative in their specialty. It also keeps us agile—when new challenges arise (e.g., optimizing the ML pipeline), the respective domain lead steps forward, with others offering support.

Overall, this flexible leadership style ensures that each person can leverage their strengths, while the team benefits from a shared sense of responsibility and balanced decision-making.

8. References

[1] Meta, "How fact-checking works," Meta Transparency Center. [Online]. Available: https://transparency.meta.com/en-us/features/how-fact-checking-works/. [Accessed: 16-Nov-2024].

[2] X Help Center, "Community Notes," X, [Online]. Available: https://help.x.com/en/using-x/community-notes. [Accessed: Mar. 8, 2025].

[3] Knight Science Journalism, "Fact-checking science journalism: How to make sure your stories are true," KSJ Handbook. [Online]. Available: https://ksjhandbook.org/fact-checking-science-journalism-how-to-make-sure-your-stories-are-true/the-fact-checking-process/. [Accessed: 16-Nov-2024].

[4] J. Waterson, "Internet overtakes TV as UK's most popular news source for first time," The Guardian, 10-Sep-2024. [Online]. Available: https://www.theguardian.com/media/article/2024/sep/10/internet-tv-uk-most-popular-news-source-first-time. [Accessed: 16-Nov-2024].

[5] Mihir J, "Browser Comparison Finale Chromium-based Browsers", Medium. [Online]. Available:https://medium.com/@mihirgrand/browser-comparison-finale-chromium-based-browsers-2b6063e74165. [Accessed: 16-Nov-2024].

[6] IBM, "What is an API?," IBM Topics. [Online]. Available: https://www.ibm.com/topics/api. [Accessed: 16-Nov-2024].

[7] GeeksforGeeks, "Natural language processing overview," GeeksforGeeks. [Online]. Available: https://www.geeksforgeeks.org/natural-language-processing-overview. [Accessed: 16-Nov-2024].

[8] IBM, "What is machine learning?," IBM Topics. [Online]. Available: https://www.ibm.com/topics/machine-learning. [Accessed: 16-Nov-2024].

[9] Google, "URL," Google Support. [Online]. Available: https://support.google.com/google-ads/answer/14095?hl=en. [Accessed: 16-Nov-2024].

[10] IBM, “What is GDPR?”, IBM. [Online]. Available: https://www.ibm.com/cloud/compliance/gdpr-eu. [Accessed: 25-Feb-2025].

[11] EFCSN, "Code of standards," EFCSN. [Online]. Available: https://efcsn.com/code-of-standards/. [Accessed: 16-Nov-2024].

[12] Encyclopædia Britannica, "Artificial intelligence," Britannica, [Online]. Available: https://www.britannica.com/technology/artificial-intelligence. [Accessed: Mar. 8, 2025].

[13] Teyit, "Teyit nedir?", Teyit, [Online]. Available: https://teyit.org/nedir. [Accessed: Mar. 8, 2025].

[14] InVID Project, "InVID verification plugin," InVID Project, [Online]. Available: https://www.invid-project.eu/tools-and-services/invid-verification-plugin/. [Accessed: Mar. 8, 2025].

[15] SurfSafe, "SurfSafe: Join the fight against misinformation," Google Chrome Web Store, [Online]. Available: https://chromewebstore.google.com/detail/surfsafe-join-the-fight-a/hbpagabeiphkfhbboacggckhkkipgdmh. [Accessed: Mar. 8, 2025].

[16] Google Fact Check Explorer, "About," Google, [Online]. Available: https://toolbox.google.com/factcheck/about. [Accessed: Mar. 8, 2025].

[17] IDIR at University of Texas at Arlington, "ClaimBuster: Live fact-checking," University of Texas at Arlington, [Online]. Available: https://idir.uta.edu/claimbuster/. [Accessed: Mar. 8, 2025].

[18] Observatory on Social Media, Indiana University, "Hoaxy FAQ," Indiana University, [Online]. Available: https://hoaxy.osome.iu.edu/faq. [Accessed: Mar. 8, 2025].

[19] Ground News, "About," Ground News, [Online]. Available: https://ground.news/about. [Accessed: Mar. 8, 2025].

[20] Amazon Web Services, "Amazon RDS," AWS, [Online]. Available: https://aws.amazon.com/rds/?nc1=h_ls. [Accessed: Mar. 8, 2025].

[21] Amazon Web Services, "Amazon EC2," AWS, [Online]. Available: https://aws.amazon.com/ec2/?nc1=h_ls. [Accessed: Mar. 8, 2025].

[22] Google Developers, "Custom Search JSON API introduction," Google, [Online]. Available: https://developers.google.com/custom-search/v1/introduction. [Accessed: Mar. 8, 2025].

[23] Google Cloud, "Cloud Vision API," Google Cloud, [Online]. Available: https://cloud.google.com/vision?hl=tr. [Accessed: Mar. 8, 2025].

[24] J. Thorne et al., "Automated fact-checking: Tasks, datasets, models, and challenges," arXiv preprint arXiv:2203.05659, 2022. [Online]. Available: https://doi.org/10.48550/arXiv.2203.05659. [Accessed: Mar. 8, 2025].