DEVELOPMENT OF WEB-BASED SIGNAL ANALYSIS TOOLS FOR THE MODERNIZED NHTSA CRASH TEST DATABASE

Tanner Filben[1], James Gaewsky[1], Ryan Barnard[2], Sarah Crimmins[1], Bobby Amoroso[2], Alison Henry[2], Matthew Davis[1], Dan Parent[3], Patrick Smith[3], F. Scott Gayzik[2]
[1]Elemance, LLC, Winston-Salem, NC
[2]Wake Forest University School of Medicine, Winston-Salem, NC
[3]National Highway Traffic Safety Administration, Washington, DC

Paper Number ESV26-239

ABSTRACT

The National Highway Traffic Safety Administration (NHTSA) maintains the Crash Test Database (CTD), a repository of vehicle crash, biomechanics, and component-level test data supporting research, regulation, and consumer information programs such as the New Car Assessment Program (NCAP). To address limitations of the legacy on-premises system and desktop software tools, NHTSA initiated a comprehensive modernization of both the CTD and its companion Signal Analysis Tools software used for reviewing signals and calculating injury metrics from test data.

The CTD modernization was executed in four Agile phases. First, the legacy Oracle database and network file storage were migrated to Amazon Web Services (AWS) cloud infrastructure, replacing the public-facing website with a modern, API-driven application. Second, a secure web portal was developed to enable credentialed users to upload metadata, test data, and media files. Third, a new Crash Avoidance Database (CADB) schema was introduced to manage data from tests of advanced driver assistance systems including Forward Collision Warning, Automatic Emergency Braking, and Lane Keeping Support. Fourth, database updates expanded support for pedestrian crashworthiness testing and integrated the web-based Signal Analysis Tools (WebSAT) into the CTD cloud environment.

In parallel, the WebSAT platform was developed to replace around 20 legacy desktop programs. WebSAT enables users to access CTD data directly from a browser, perform time-history signal processing, and compute validated injury metrics relevant to crashworthiness. Automated test suites ensure back-end computations maintain consistency with biomechanics standards, Federal regulations, and legacy algorithms. Results generated in WebSAT can be viewed directly in-browser or downloaded to the user's machine for additional analysis or reporting.

Together, the modernized CTD (https://www.nhtsa.gov/research-data/research-testing-databases/) and WebSAT platform (https://websat.nhtsa.dot.gov) provide an accessible framework for test data storage, analysis, and visualization. These efforts improve data integrity, reproducibility, and efficiency, advancing both NHTSA's and the public's ability to evaluate and communicate vehicle crashworthiness.

Keywords: crash test database, signal analysis, biomechanics, injury metrics, crashworthiness

INTRODUCTION

In September of 1966, the National Traffic and Motor Vehicle Safety Act ("Safety Act") (49 U.S.C. Chapter 301) [1] was signed into law in the United States. The Safety Act directs the Secretary of Transportation to establish appropriate Federal Motor Vehicle Safety Standards (FMVSS) that would lead to the reduction of the number of deaths and injuries resulting from motor vehicle accidents. Each FMVSS must be practicable, meet the need for motor vehicle safety, and be stated in objective terms. In prescribing standards, the Secretary must consider: (1) relevant motor vehicle safety data, (2) whether the proposed standard is reasonable, practical, and appropriate for the particular type of motor vehicle equipment for which it is prescribed, and (3) the extent to which such standards contribute to carrying out the purposes of the Act.

To meet the above requirements, the National Highway Traffic Safety Administration (NHTSA) is directed to develop safety standards. When proposing new FMVSS, NHTSA often undertakes extensive research to ensure that the proposed standard satisfies the requirements of the Act. NHTSA also conducts a variety of research to support other agency objectives, such as research needed to support the New Car Assessment Program (NCAP). NHTSA established NCAP in 1978 in response to Title II of the Motor Vehicle Information and Cost Savings Act of 1972. NHTSA currently requires vehicle manufacturers to include safety rating information, obtained from NHTSA under its NCAP program, on the Monroney labels of all new light vehicles manufactured on or after September 1, 2007 (49 CFR part 575). This requirement was mandated by Section 10307 of the Safe, Accountable, Flexible, Efficient Transportation Equity Act; A Legacy for Users (SAFETEA-LU). The purpose of the law is to ensure that vehicle manufacturers provide consumers with relevant vehicle safety ratings information on all new light vehicles at the point of sale so that they can make informed purchasing decisions.

For example, for crashworthiness research, NHTSA often conducts crash testing, which involves collecting data from various transducers mounted to the test dummies, vehicles, and motor vehicle equipment, recording high-speed films or videos of the event, taking pictures of the test setup, and generating a written report. Since 1978, these data have been loaded into a single data repository, where NHTSA staff and the public can access the data and conduct analyses. This repository is known as the NHTSA Crash Test Database (CTD). The data stored in the CTD has been used to support numerous rulemakings over the years, satisfy Freedom of Information Act (FOIA) requests, and support research conducted by university [2], industry [3], and government [4] entities.

The legacy CTD included three database schemas: Vehicle Crash Test Database (VehDB), Biomechanics Test Database (BioDB), and Component Test Database (ComDB). The VehDB contains engineering data measured during various types of research, NCAP, and compliance tests typically involving full-vehicle crashes. VehDB tests can include multiple vehicles and occupants per database entry and include an expanded set of metadata describing the vehicle(s) involved in the crash test. The BioDB is a repository of experimental data used in the development and evaluation of crash testing tools such as Anthropomorphic Test Devices (ATDs) and human body models (HBMs) as well as related injury criteria, typically involving sled tests or isolated impacts using specialized test configurations. BioDB tests include only one occupant per database entry and include an expanded set of metadata describing the occupant. The ComDB was designed to hold data associated with multiple tests or impacts grouped into a single database entry. Examples include motorcycle helmet testing and interior impacts.

Currently, the most popular and critical use of the CTD is to store data used in the calculation of vehicle ratings carried out for NCAP. NHTSA performs crashworthiness and crash avoidance tests at several test laboratories and stores the resulting data, photos, videos, and reports in the CTD. The data from these tests are then processed to calculate star ratings, which are subsequently displayed on the NHTSA website and the vehicle's Monroney label to provide consumers with comparative safety information to assist in vehicle purchase decisions.

Until recently, the legacy CTD was hosted on information technology (IT) infrastructure located physically within the Department of Transportation headquarters. It was accessible internally through direct Structured Query Language (SQL) database queries to identify tests of interest, and access to related data and media files on a shared network drive. It was also accessible to the public through a web application which allowed search capabilities, displayed test information and photographs, and allowed download of media files and data files in several formats. NHTSA published Test Reference Guides (TRGs) describing the structure of the repository and data submission requirements.

NHTSA also published a set of data analysis programs, collectively known as the NHTSA Signal Analysis Tools, which facilitated viewing and post-processing of the data stored in the CTD. These programs included tools for viewing time-history data (PlotBrowser, CompareSignals), signal processing (BwFilt, Fir100, Resultant), and injury criteria calculation (HIC, Nij, Clip3ms, etc.). NHTSA also developed a program known as Entrée, which assisted researchers in formatting information describing CTD tests consistent with the TRG (Figure 1). These tools were originally developed to be compatible with 32-bit Windows operating systems but were later updated to support 64-bit Windows operating systems. In total there were more than 30 standalone applications facilitating signal analysis, data loading and publishing, and NCAP star rating generation.

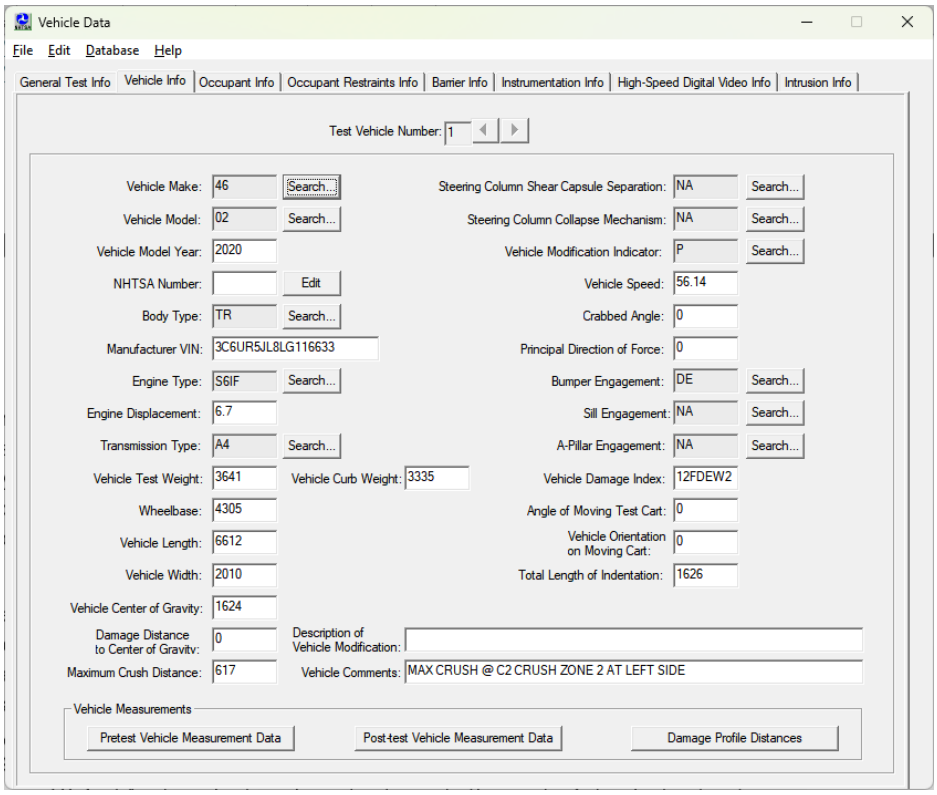

Figure 1. Legacy Entrée software interface.

To improve quality of service, prevent loss of data, and conform with modern information technology usability and security practices, an effort to modernize the CTD was initiated in Spring 2021. The goal of this project was to migrate the CTD database and web application to cloud infrastructure. In the process, a web-based version of the Entrée program was developed to facilitate the upload of test information, test data, and media files by the facilities conducting the crash tests, subsequent review by NHTSA staff, and release of these tests to the public.

The modernization of the CTD also provided an opportunity to introduce new capabilities. One new feature was the addition of a Crash Avoidance Database (CADB). Before the CADB was developed, information on the crash avoidance testing was stored in the VehDB. Unfortunately, the existing database table structure and specific fields in the VehDB would not allow for automated processing of CADB test results. Additionally, the web application front end for the VehDB was not well suited for the display of crash avoidance information. An additional feature added in the process of modernizing the CTD was the expansion of the ComDB to enable storage and display of pedestrian crashworthiness testing data. This involved the addition of several new database tables and fields to store test parameters specific to pedestrian crashworthiness testing and enable automated processing of results.

Once the cloud migration was complete, a parallel effort was initiated to modernize the Signal Analysis Tools. Since these tools needed to be updated to interface with the modernized database, NHTSA used this opportunity to modernize the tools themselves to improve stability and usability. While the CTD modernization project was implemented as a competitive award under an IT shared services contract, the proposal for the Signal Analysis Tools

effort was solicited under the Experimental and Mathematical Biomechanics Injury Research (EMBIR) contract in order to involve the researchers that were already familiar with the use of the signal analysis methods and injury criteria calculations.

The objective of this manuscript is twofold: (1) detail the efforts to modernize the CTD and (2) describe the development of a unified, browser-accessible signal analysis tools platform that interfaces directly with the CTD, automates validated injury metric and crashworthiness computations, and improves reproducibility and efficiency for safety engineers.

METHODS

Modernization of the CTD

The modernization of the CTD was implemented in four phases following an Agile project management framework to carry out the planning, design, development, testing, deployment, and review. The four phases are summarized below:

Phase 1 - Cloud migration: In this phase, the on-site relational database was migrated from Oracle to Amazon Web Services (AWS) RDS PostgreSQL, network file storage was migrated to AWS S3 storage buckets, and the public-facing website [5] was redeveloped using modern web technologies. The latter step included development of an Application Programming Interface (API) which allowed structured communication between the display layer and the database layer [6–9].

Phase 2 - Data loading portal: In this phase, a web application was created to allow government staff and test laboratory personnel to upload metadata, data, photos, videos, and reports to the cloud infrastructure developed in Phase 1. This web portal was designed to incorporate credentialed access enabled using the Login.gov framework for external users and organizational single sign-on (SSO) authentication for internal users. Previously, test laboratories would either mail CDs, DVDs, or portable hard drives or use FTP or FTP-equivalent web services to transfer the files to NHTSA contract staff, often resulting in delays, lost files, and large-scale disorganization. Additionally, NHTSA CTD submissions previously required a proprietary “EV5” header file that was not user friendly (Figure 2), so effort was made to support JSON file formats. JSON is an open standard file format that can be generated and parsed by many modern programming languages.

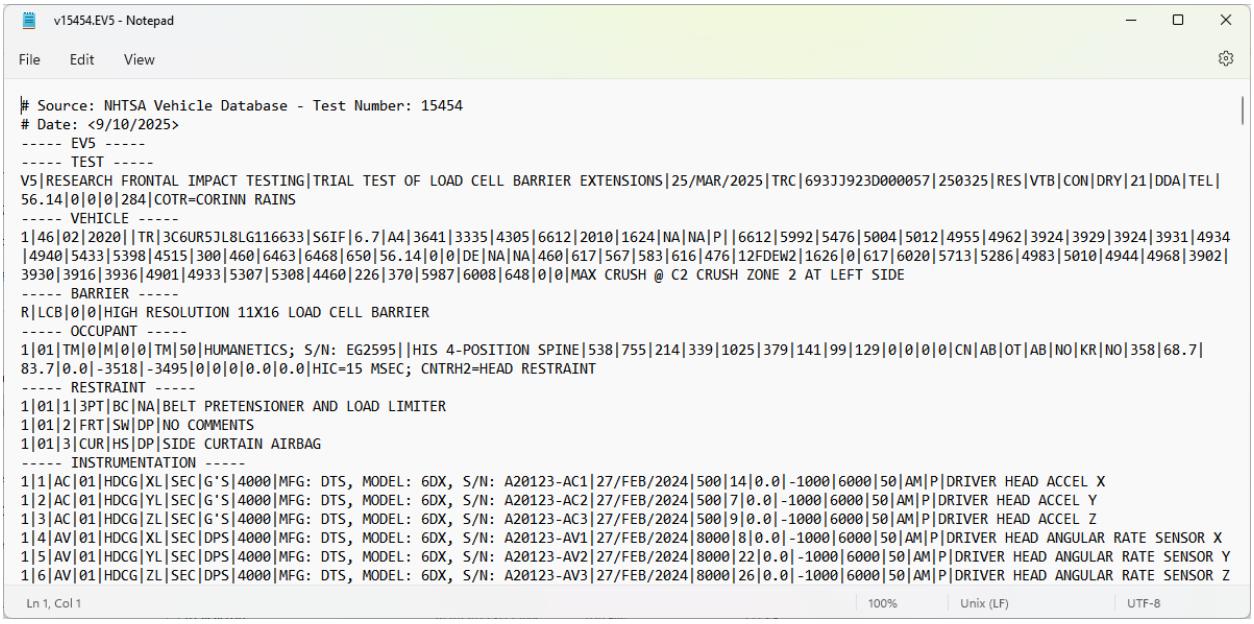

Figure 2. Legacy NHTSA EV5 file format.

Phase 3 - Crash avoidance database (CADB): Phase 3 of the CTD modernization process aimed to add a new schema to the relational database tailored to the storage and display of data from tests of vehicle crash avoidance

technologies, including Forward Collision Warning (FCW), Crash Imminent Braking (CIB), Dynamic Braking Support (DBS), Pedestrian Automatic Emergency Braking (PAEB), Lane Departure Warning (LDW), Lane Keeping Support (LKS), Blind Spot Warning (BSW), and Blind Spot Intervention (BSI).

Phase 4 - Completion of modernization: The final phase of the CTD modernization project included several important updates that were not addressed in previous phases, but necessary to complete the modernization. This included updates to the Vehicle, Biomechanics, and Component databases to leverage the enhancements implemented in Phase 3 to remove dependencies on on-premises software by January 2026. In the process, the Component database was updated to "version 6" and expanded in September 2025 to enable storage and display of data from pedestrian crashworthiness tests, such as those described in NHTSA's September 19, 2024 notice of proposed rulemaking that proposed new requirements for pedestrian head protection (89 FR 76922) [10] and NHTSA's November 25, 2024 final decision notice on crashworthiness pedestrian protection for NCAP (89 FR 93000) [11]. Another portion of Phase 4 aims to incorporate the modernized Signal Analysis Tools web application into the existing cloud infrastructure in early 2026.

Web-based Signal Analysis Tools (WebSAT) Architecture and Technology Stack

Overall design principles: WebSAT is a containerized, horizontally scalable application with only ephemeral application state. The core application is a Django-based [12] backend service that exposes the full functionality of the signal analysis tools as API endpoints. Users interact with WebSAT via a Vue.js-based [13] single-page application (SPA) web interface that is a client of that API. This API-first approach ensures that the full functionality of the application is supported for both web users and advanced users who need to build custom functionality for novel use cases.

WebSAT's implementation is divided into three code repositories: WebSAT-Signal, WebSAT-App, and WebSAT-Infra. The WebSAT-Signal repository is a Python library that implements the full suite of signal operations and injury metrics. As WebSAT-Signal is fully independent of the web application, it can be imported as a project dependency in other, potentially non-web-based projects, expanding the set of potential use cases for the signal analysis tools. The project's documentation, which uses the Docusaurus [14] static documentation website generator, is contained in the WebSAT-Signal repository.

The WebSAT-App repository houses the Django backend project as well as the front-end implementation. It is the primary consumer of WebSAT-Signal, which it imports as a dependency, thus embedding the signal analysis implementations within the application. The WebSAT-App repository also contains the project's Dockerfile, which creates a container image consisting of both the Django application and the compiled JavaScript and CSS bundles comprising the front end. These bundles are copied into the Django application's static assets folder and served directly to end users via WhiteNoise [15].

The WebSAT-Infra repository contains a complete set of OpenTofu [16] modules and services for deploying staging and production instances of the WebSAT application on AWS [17].

At runtime, a minimal deployment of WebSAT consists of three containers: (1) the WebSAT-App application container, which serves the front-end JavaScript and CSS bundles and exposes the signal analysis computations via its API, (2) a Celery [18] worker container, which performs long-running computations and asynchronous tasks, and (3) an instance of the Redis [19] key-value cache for coordination between the application and Celery containers and for caching of remote resources and the signal computation results. At build time, the documentation is rendered as a static website and can be hosted on any web server.

Development of signal analysis calculations (WebSAT-Signal): The WebSAT-Signal repository was developed to modernize and consolidate the numerical computation functions previously distributed across more than 30 standalone desktop applications within the legacy NHTSA Signal Analysis Tools suite. Its primary purpose is to provide validated and version-controlled implementations of injury metric and crashworthiness calculations consistent with the analytical methods historically used in support of FVMSS, NCAP, and other agency research.

The WebSAT-Signal backend was implemented in Python to promote transparency, maintainability, and interoperability with the modernized CTD. Each calculation module reproduces the algorithms of the legacy desktop

programs using standardized digital signal processing procedures (e.g., channel filtering, sampling, differentiation) consistent with regulatory definitions and SAE International practices.

Validation of each module followed a two-tier process. First, numerical outputs from WebSAT-Signal were compared against results generated from the corresponding legacy desktop tools. Automated test suites were implemented to ensure that computed values matched reference results within defined numerical tolerances. Second, the unit tests were incorporated into a continuous integration pipeline to verify conformance following any code modification. Discrepancies were reviewed against legacy application source code, SAE protocols, and NHTSA documentation to ensure calculated results were computed using the most current recommendations or regulations.

All validated functions are maintained under version control and corresponding documentation. This ensures reproducibility and transparency for regulatory analyses and public use. The validated WebSAT-Signal repository serves as the computational core for the WebSAT web application, providing consistent calculations across the datasets accessible through the modernized CTD.

WebSAT front end: The front end of WebSAT is a SPA built on Vue.js, with PrimeVue [20] used for many of the UI components. WebSAT uses the SPA pattern because the application is meant to present an integrated data analysis environment where separating views and components onto separate pages would be a hindrance to a smooth user experience. Vue was chosen for its low learning curve and the balance between reactivity (i.e., UI components that update automatically when application state changes) and explicitness (i.e., the ability for a programmer to directly control data flow and program behavior without relying on hidden or implicit framework behavior). PrimeVue's accessibility-compliant, off-the-shelf UI components provided significant support for meeting Section 508 compliance application.

WebSAT uses ESBuild [21] for rapidly building the front-end application bundle, Yarn [22] for managing dependencies, and Jest [23] for unit testing. Initially written in JavaScript, much of the front end is undergoing a gradual migration to TypeScript for stronger type safety and improved maintainability.

The JavaScript and CSS bundles comprising the front end are embedded in the Django application's static assets folder during build; the application serves those files using the WhiteNoise library. This simplifies deployment and reduces the disparity between development and production infrastructure. Should traffic scale to the point that serving static assets directly from the application were to become a bottleneck, the application can be easily reconfigured to front those assets with a Content Delivery Network like CloudFront just by adjusting one environment variable.

WebSAT back end: WebSAT's back end is built around the Python-based Django web framework. Django is a mature project with a strong security posture that promotes building maintainable software architecture. The Django Rest Framework (DRF) [24] provides clean patterns and plumbing for supporting WebSAT's API-first design philosophy, with drf-spectacular [25] facilitating generation of OpenAPI 3-compliant schema files and user-friendly Swagger UI [26] and Redoc [27] web interfaces.

Many of the operations that WebSAT supports require non-trivial amounts of time (due to dependence on the upstream NHTSA CTD API) or computation to complete. As such, most client requests, such as loading instrumentation time series data, performing operations on signal data, and computing injury metrics, do not produce immediate results and are instead computed asynchronously. The back end responds with a 202 (Accepted) status and a unique "task identifier" associated with the request. Clients can retrieve the result of the computation by polling a special "task result" endpoint, the URL of which is provided in the Location header of the initial server response. To improve performance and reduce the number of distinct server connections the polling strategy requires, WebSAT supports WebSockets (via the Daphne [28] library) in addition to traditional HTTP (non-WebSocket) connections. When first loaded, the front-end code attempts to establish a long-lived WebSocket connection, but if that connection fails or is lost, it gracefully falls back to standard HTTP connections and the polling strategy. Should a lost WebSocket connection become re-established, the front end will automatically resume sending requests and awaiting responses via the WebSockets.

These asynchronous operations are supported by the Celery task queue: when a request for data that must be handled asynchronously arrives, the back end validates the request, then submits that request to Celery, using Redis as a

message broker. Enqueueing new tasks is an inexpensive operation, which allows the back end to respond quickly to requests. Alongside WebSAT's Django application, one or more Celery "worker" processes run in another container or on another server. When a new task is posted to the Redis message queue, an available worker inspects that task and retrieves the requested data or performs the requested computation, then posts the result back to Redis. Clients with WebSocket connections are immediately notified of the new result, while clients using the polling strategy will obtain that result upon their next query of the task result endpoint.

All requests are idempotent, and identical requests from different clients are intended to yield identical results. As such, if Client A makes a request, and Client B makes the same request before the result from Client A's request has completed, Client B is assigned the same task identifier as Client A and is notified of the result at the same time as Client A (i.e., the back end does not perform the computation twice). Those task results are also retained in the Redis cache so that, if Client C eventually makes the same request, the previously computed and cached result is returned without performing the computation again. However, no cache can store an infinite amount of data, and some requests and results must inevitably be evicted. When this happens, WebSAT is able to detect that a task result is no longer available and will recompute missing data as needed.

Thus, unlike most web applications, WebSAT has an ephemeral computation model where all results are transient, re-computable, and do not require long-term storage. As such, Redis is the application's only persistence layer, and does not need to be configured to persist any data to disk.

WebSAT computation model: A common use case in WebSAT is the need to "compose" two or more functions. For example, a user may wish to perform a sequence of operations such as:

1. Load the time series data for three instrumentation channels,
2. Apply a Butterworth filter to those signals with a particular cutoff frequency,
3. Generate the resultant of those signals,
4. Load the time series data for another instrumentation channel, and
5. Create an X-Y plot of the resultant from step 3 and the unmodified time series data from step 4.

Each step can be modeled as a function with specific inputs and outputs. For example, loading time series data is a function, f, with the instrumentation channel's identity as its input and the time series data as its output; filtering a signal is a function, g, with a signal and bandwidth as its inputs and a new time series signal as its output. Since the filtering operation g requires the output of the loading operation f as an input, the second step is essentially the composition of g and f, $g(f(x))$. Each of the other steps can likewise be represented as functions with one or more inputs and outputs.

However, as a practical consideration in the context of a web-based software application, it is not always desirable for the full input data of every function the user invokes to be transmitted from the web browser to the application server. In the case of loading time series data, the function's input is a simple set of identifiers and is trivial to upload. As a matter of course, the function's output (i.e., the time series signal itself), needs to be downloaded from the server to the browser for it to be displayed to the user. But the input to the subsequent filter function is that same time series signal: in some cases, this can be a substantial amount of data, and most users are on internet connections that cannot upload data to a server as quickly as data can be downloaded. Furthermore, that time series data is the same information that was just transmitted from the server to the user; sending that data right back to the server is a redundant doubling of traffic.

In WebSAT, each function that the user can perform produces, along with the function's normal output, a provenance token, which is a compact identifier encoding both the operation and its inputs. When the user wishes to apply the filter function to a signal that was just downloaded, the front end need only supply that brief provenance token rather than the entire signal. As each operation produces another provenance token, complex analyses can be built compositionally without resending large datasets.

In addition, when WebSAT generates a provenance token for an operation, that operation's result is stored in the Redis cache, keyed by the provenance token. This ensures that the results of previous operations can be inexpensively reused for multiple subsequent tasks without requiring re-computation.

Because cached results will inevitably become evicted from the cache due to age or memory pressures, provenance tokens are invertible (i.e., they contain sufficient information to fully recompute the data they represent from the very first inputs provided by the user if necessary). In practice, it is not always necessary to fully recompute a result: if the data associated with the composition *h(g(f(x)))* has been evicted from the cache but *g(f(x))* is still present, the intermediate value *g(f(x))* will be reused and only the function *h* will be recomputed. Through this mechanism, WebSAT is able to perform the minimum amount of necessary work to re-generate complex composition chains in which only a subset of the constituent results has been evicted from the cache.

Infrastructure and deployment: WebSAT was designed to be agnostic to its deployment environment, and it can run as native services on one or more hosts. As it has no baked-in dependencies on third-party runtime services outside of the NHTSA CTD API, WebSAT is also cloud-agnostic and can run either on premises or on infrastructure owned by any cloud provider.

To ease adoption by NHTSA, a demonstration instance of WebSAT was deployed on AWS, with the complete infrastructure stack—including separate staging and production environments—provided as a set of OpenTofu Infrastructure as Code (IaC) modules and services. This demonstration instance is a containerized deployment using AWS Elastic Container Service (ECS), using plain EC2 instances; compatibility with Fargate has also been confirmed. Application Load Balancer (ALB) fronts the WebSAT backend service and secures traffic using a TLS certificate issued and managed by AWS Certificate Manager (ACM). Though WebSAT has no integrated mechanism for authentication or authorization, the application is protected from unauthorized access during the development process via Cognito; users who are eligible to access WebSAT are sent invitation e-mails through Simple E-mail Service (SES). SES is also used for sending automated notifications of errors to project managers. The static documentation site is hosted from an S3 bucket and distributed via CloudFront.

GitHub Actions orchestrate automated testing of code in the WebSAT-Signal and WebSAT-App repositories. When code is pushed to the main branch of WebSAT-Signal, it also triggers the documentation site to be built and published to S3, followed by invalidation of the CloudFront distribution. The WebSAT-App repository has workflows for building the application container image and pushing it to AWS Elastic Container Registry (ECR). Pushes to WebSAT-App's staging branch additionally trigger an update of the staging ECS cluster's task to reference the new container image; pushes to the main branch—which are protected by requiring approval from at least two developers—do the same for the production cluster.

RESULTS

NHTSA CTD Updates

Phase 1 - Cloud migration: The technical architecture of the modernized CTD is described in Appendix A (Figure A1). The two key viewports in the public-facing website are the search page, which facilitates querying using common descriptive fields, and the test detail page (Figure 3), which summarizes individual tests, displays relevant photos and videos, and allows download of data, reports, photos, and videos from each test. The search page and the test detail page are customized for each database schema. Phase 1 was completed in January 2022.

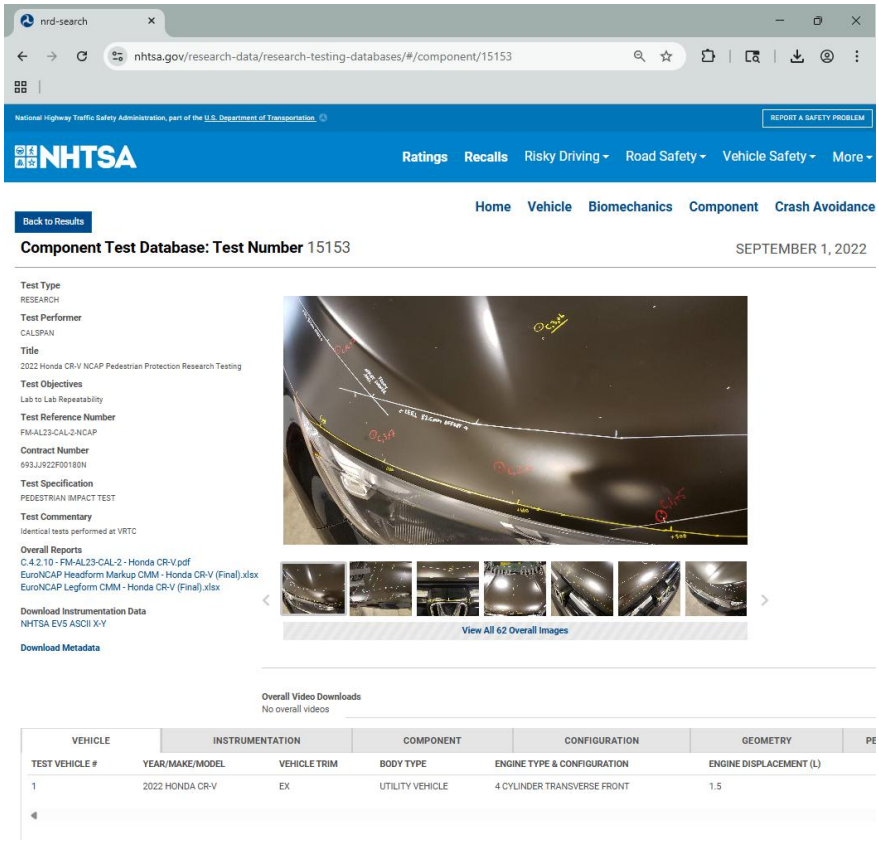

Figure 3. Test detail page of the modernized CTD.

Phase 2 - Data loading portal: Uploads to the new portal are organized by database, filterable, and sortable (Appendix B, Figure B1). Users can also opt to only see tests they submitted or that have been assigned to them for review. The new portal also introduced the ability to upload and download test metadata in a standardized JSON format. However, at the completion of Phase 2, the process of releasing this uploaded data to the public was still dependent on a series of programs and scripts built on on-premises systems. The data portal allows users the option to either create a submission entirely through the user interface (Figure 4) or the ability to populate the fields through the upload of header data in the EV5 or JSON format. The user interface includes validation rules for ensuring entered data matches requirements such as upper and lower limits, dependencies, field types, and mandatory data. There are some additional useful features such as auto-calculated fields and a tie into NHTSA's vPIC VIN decoder tool that will auto-populate vehicle information when a valid VIN is entered by the user [29]. Furthermore, all signal data and media can be zipped and bulk uploaded in a single step. Phase 2 was initially completed in September 2023 with expanded functionality added in subsequent phases.

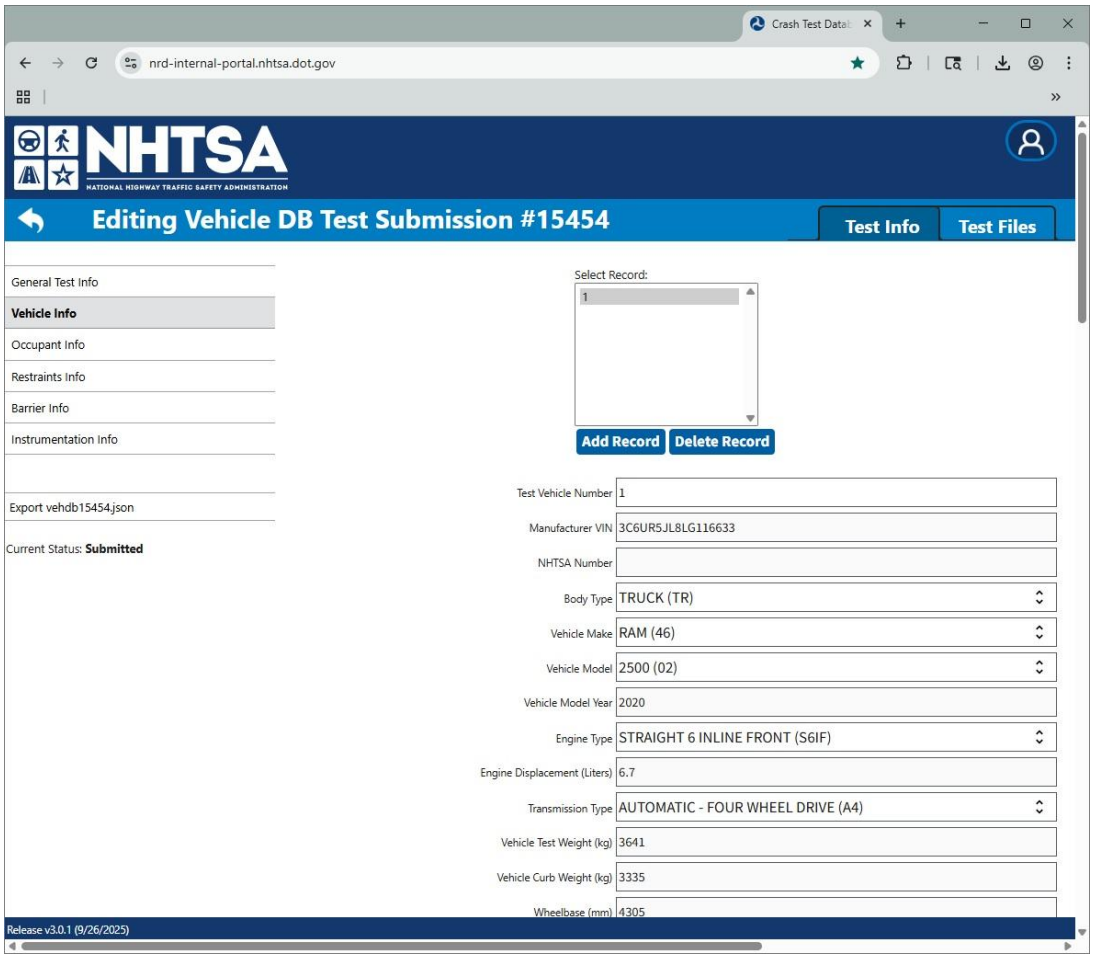

Figure 4. Modernized CTD portal interface for submission of test data.

Phase 3 - Crash avoidance database (CADB): The public-facing website and data loading portal from the previous two phases were updated accordingly to incorporate updates specific to the CADB. As the CADB was an entirely new schema [30] with no legacy components (Appendix C, Figure C1), development of Phase 3 was carried out such that no on-premises programs or scripts were necessary for intake, processing, and release of test data to the public website. The initial use of the new CADB is to support NCAP crash avoidance testing as outlined in 89 FR 95916 [31]. During Phase 3 a Jira-based help desk system was created for internal and external users to submit tickets to the development, modernization, and enhancements (DME) and operations and maintenance (O&M) team to request help, report bugs, and offer feature requests. Phase 3 was completed in November 2024.

Phase 4 - Completion of modernization: At this stage of Phase 4 development NHTSA is able to intake data from internal and contract test laboratories, review and approve data submissions, and publish data to the public website without the need for contractor personnel or on-premise servers. NHTSA is also able to provide new internal and external users access and assign roles with varying permission levels without the need for contractor personnel. The final remaining portion of Phase 4 is to update the BioDB and VehDB schemas to "version 6," similar to what was done to ComDB in September 2025. Version 6 will incorporate new tables, variables, and user test reference guides to ensure BioDB and VehDB are optimized for the intake of test data and media for years to come.

WebSAT Overview

The NHTSA WebSAT user interface design (Figure 5) includes a top toolbar for operations like (A) loading test data from the NHTSA CTD, (B) exporting results to a PDF report, and (C) linking users to WebSAT documentation, the NHTSA CTD, and the CTD API. From the "Load Test" section (A), users can load data from one or more tests directly from the CTD into the WebSAT interface. Upon loading test data, the main content area is populated with

metadata about the test, (D) a signal preview tool, and (E) an interactive and detailed list of all the instrumentation channels for the loaded test(s). This enables users to review signal(s) from within a single test or compare across multiple tests.

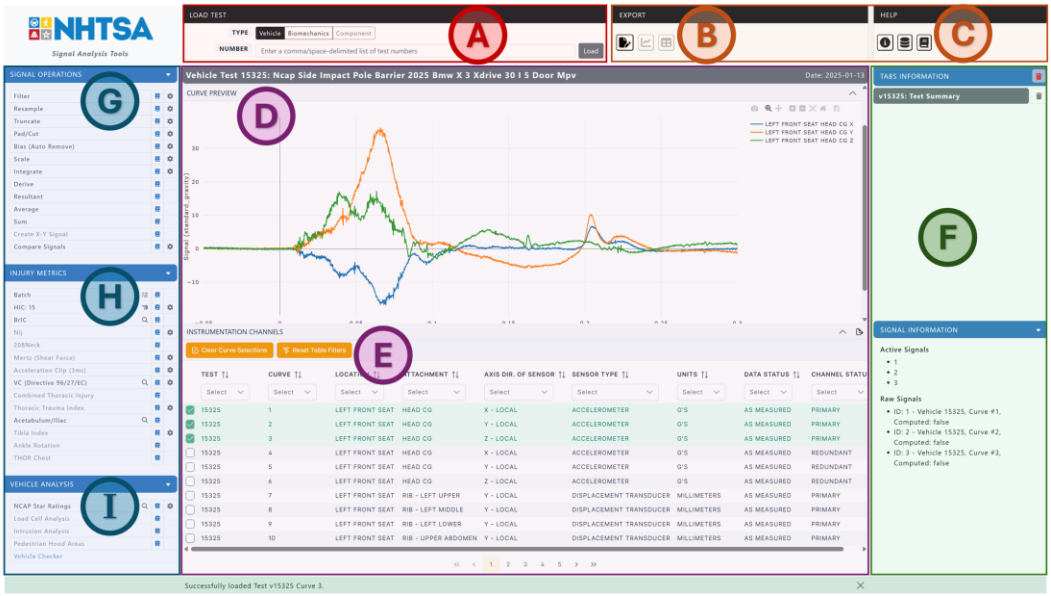

Figure 5. Overview of the WebSAT layout.

The right-side panel (F) summarizes the active tabs (i.e., which test data or results contexts are being actively displayed) and signal information (i.e., which time history signals are currently displayed). The tabs on the right side are categorized by the test or group of tests to which they belong. A new tab is generated for each test or group of tests that is loaded. Similarly, a new tab is generated for each injury metric calculation that is performed.

The left side panel consist of three main sections: (G) a Signal Operations sections that allows users to modify the raw signals loaded from the NHTSA CTD, (H) an Injury Metrics section that provides users the ability to perform biomechanical analyses of injury risk across multiple body regions, and (I) a Vehicle Analysis section that provides users tools for vehicle-centric analyses, such as generation of NCAP star ratings.

Additionally, a feature to batch process injury metrics for a given test or group of tests is available in the “Batch” interface Figure 6. This tool automatically processes all the summary injury metrics, injury assessment reference values (IARVs), and injury risks available across the requested injury metrics.

The following sections detail example analyses and results generated by WebSAT.

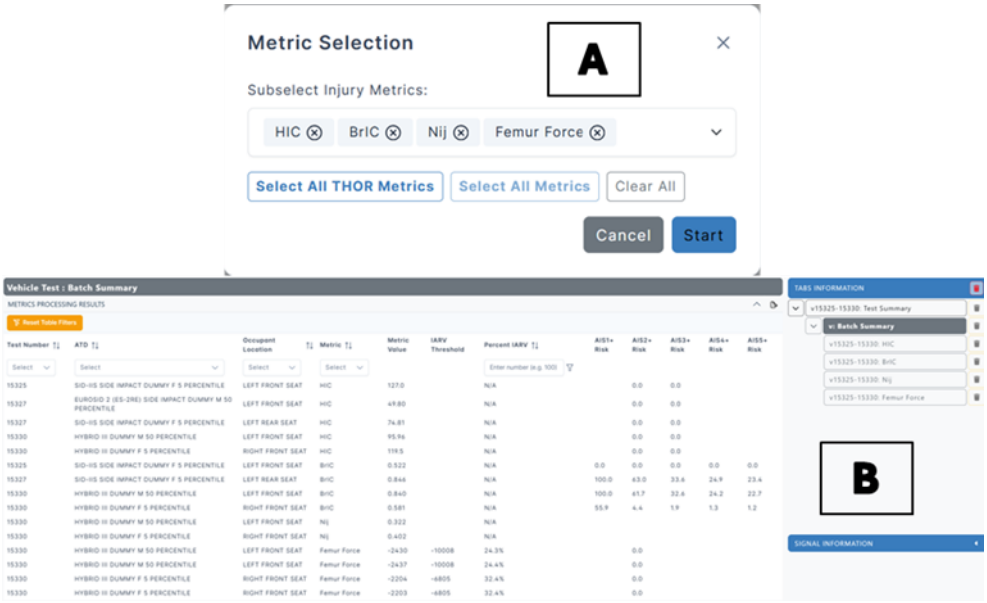

Figure 6. Batch processing metric selection and results interface.

Signal operations: After the desired test or set of tests is loaded, and a specific signal is selected from the available instrumentation channels, the Signal Operation functions become available. These functions allow the user to modify and analyze the displayed curves within the signal preview tool (Figure 7). These signal processing calculations include tools to (1) filter, resample, truncate, pad and cut, bias / debias, and scale individual signals, and (2) calculate integrals, derivatives, resultants, averages, or sums from multiple signals. There are also tools to cross-plot two time history signals (e.g., create a force vs. deflection plot) or perform quantitative comparisons of two or more time history signals. For each operation, the user can specify the appropriate parameters (via the settings icon) to ensure the signal is modified according to the desired analysis requirements.

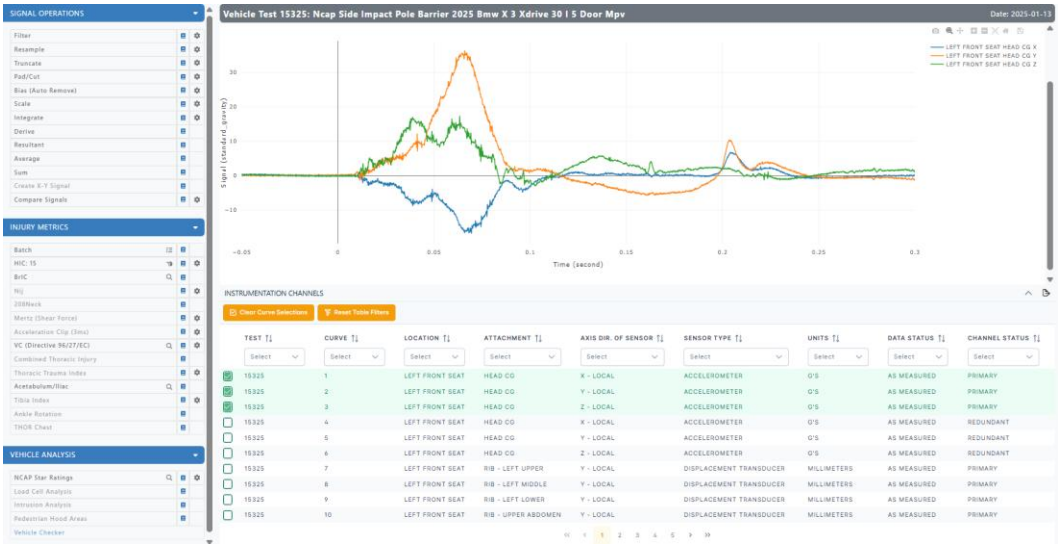

Figure 7. WebSAT signal preview interface.

Injury metrics: When a test or set of tests is loaded, the buttons for injury metrics that can be calculated from the available test data automatically become enabled. If a user clicks a specific injury metric button, WebSAT will automatically identify and retrieve the relevant signals for each occupant and perform the desired calculations. Alternatively, users can specify the exact signals they wish to use to perform the calculation by selecting the checkboxes in the instrumentation table. Some metrics have additional configurations that can be set by the user via

the settings icon adjacent to the injury metric button. The complete list of injury metrics currently supported in WebSAT are shown in Table 1.

Table 1.
Overview of occupant injury metrics available in WebSAT

Region	Injury Metric	Description
Head	Head Injury Criterion (HIC)	Calculates HIC for window sizes 15 ms or 36 ms, Skull Fracture Correlate (SFC), and a 3D plot to analyze different window sizes
	Brain Injury Criterion (BrIC)	Calculates BrIC and the corresponding risk value for AIS 1+ - AIS 5+
Neck	Neck Injury Criterion (Nij)	Calculates Nij, Ntf, Nte, Ncf, and Nce
	208 Neck	Calculates peak values for axial force, shear force, and bending moment
	Mertz	Compares peak values of axial force, shear force, or bending moment to the injury risk reference curves
Thorax	Acceleration Clip (3ms)	Calculates 3ms clip for a given signal
	Viscous Criterion (VC)	Calculates VC with the users chosen method (SAE J1727 1, SAE J1727 2, Directive 96/27/EC, or SID 2S) to understand chest injury
	Combined Thoracic Injury (CTI)	Calculates the CTI by using chest acceleration and chest deflection
	Thoracic Trauma Index (TTI)	Calculates the TTI for side impacts by using rib and spine accelerations
	THOR 3D Deflection	Visualizes the THOR-50M and -5F IR-TRACC deflections
Pelvis and Lower Extremity	Acetabulum/Iliac	Calculates pelvis force for side impact dummies
	Tibia Index (TI)	Calculates TI for the upper and lower tibia load cell
	Ankle Rotation	Calculates ankle rotation in the x, y, and z axes

Upon completion of a requested injury metric calculation, WebSAT returns the results in both tabular and graphical formats. Most injury metrics will return a summary table containing relevant scalar summaries along with one or more figures of the processed time history signals and/or relevant risk curves. Both tabular and graphical results can be saved directly to the user's computer (e.g., as CSV or PNG files) or can be copied to the clipboard to be pasted elsewhere. For example, the Head Injury Criterion (HIC) function calculates the maximum weighted integral of the resultant head linear acceleration over a specified time window. The function outputs the HIC value, the Skull Fracture Correlate (SFC), and a three-dimensional plot illustrating the results across multiple window durations. An example of the HIC graphical results for vehicle test number 10099 is provided in Figure 8.

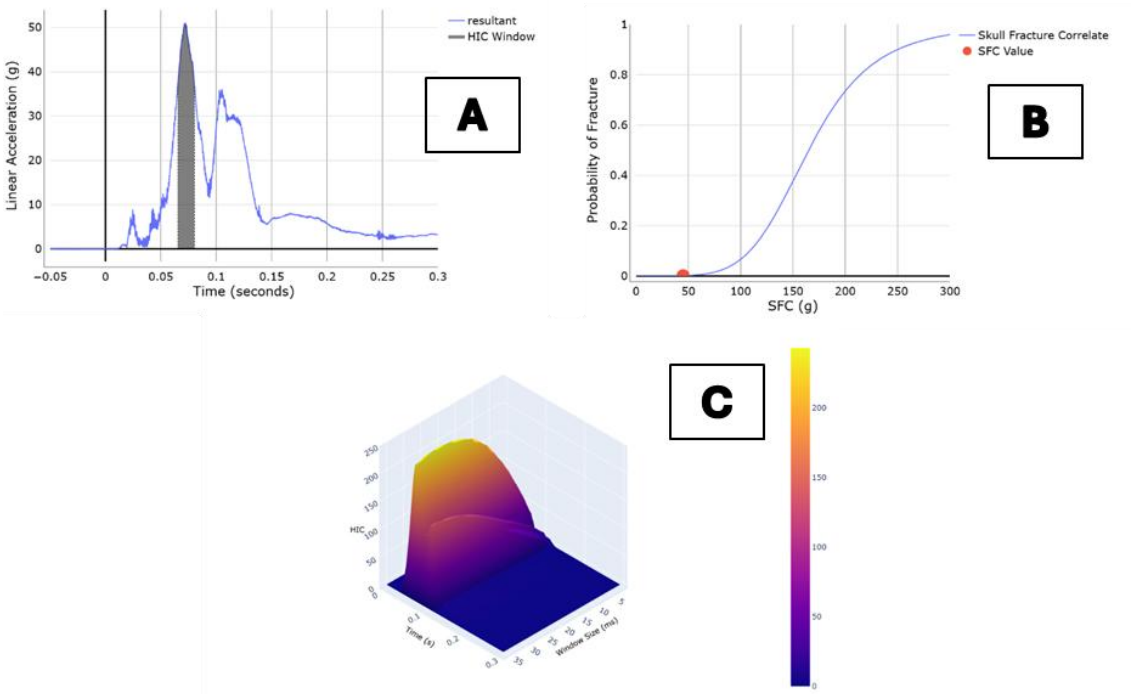

Figure 8. Sample results for a HIC calculation, including (A) the HIC window, (B) the skull fracture correlate risk curve, and (C) a three-dimensional surface plot of HIC values.

Similarly, the Neck Injury Criterion (Nij) function evaluates neck injury risk based on the combined effects of axial tension or compression and flexion or extension moments applied to the neck. Each polarity condition is analyzed independently to assess injury potential in each direction of neck motion (Tension–Flexion, Compression–Flexion, Tension–Extension, and Compression–Extension). Time-history curves for each polarity are plotted to illustrate their respective contributions to the overall Nij calculation. The occipital condyle moment and axial load are displayed with their corresponding limit curves to characterize the neck loading mode. Additionally, standard time-history plots of axial load and bending moments are provided to enable comprehensive evaluation of the neck loading response. An example of the Nij results for vehicle test number 10099 is provided in Figure 9.

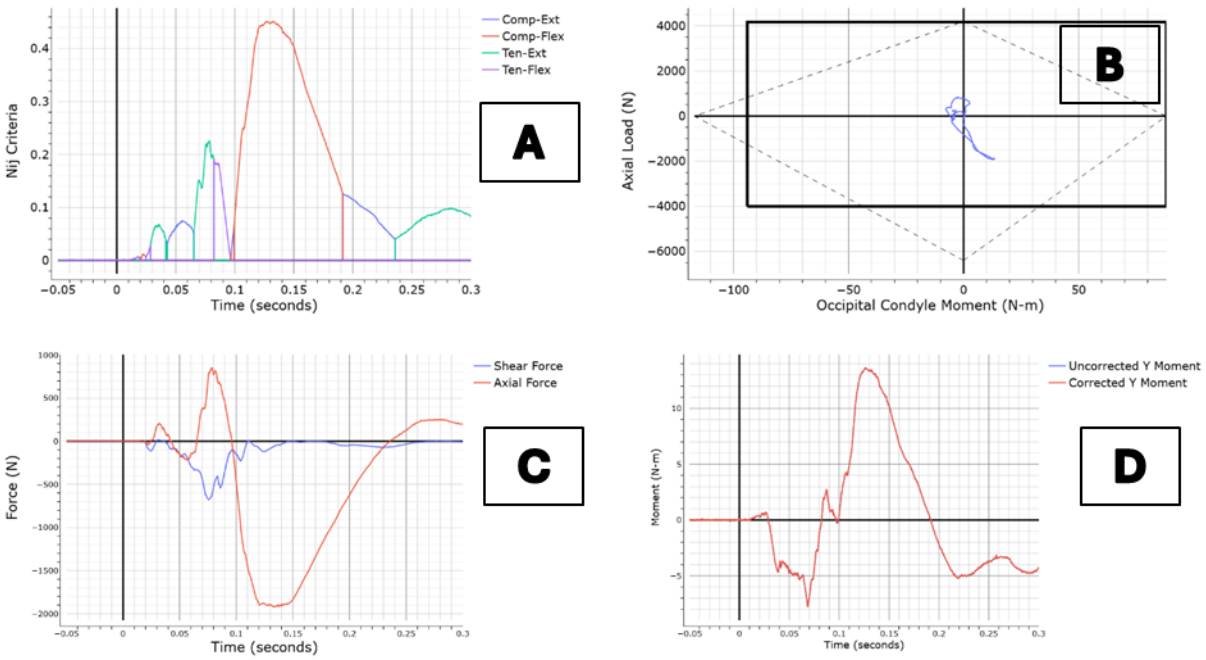

Figure 9. Sample results of an Nij calculation, including the (A) Nij time history, (B) kite plot, (C) neck force time histories, and (D) neck moment time histories.

As a final injury metric example, the THOR 3D Deflection tool is used to analyze the displacement of the Infra-Red Telescoping Rod for the Assessment of Chest Compression (IR-TRACC) sensors within the THOR-50M and -5F ATDs. The visualization displays the sensor locations along with their original reference positions, enabling the user to observe and quantify chest motion throughout the crash event. An example of the THOR 3D Deflection results for vehicle test number 10099 is provided in Figure 10.

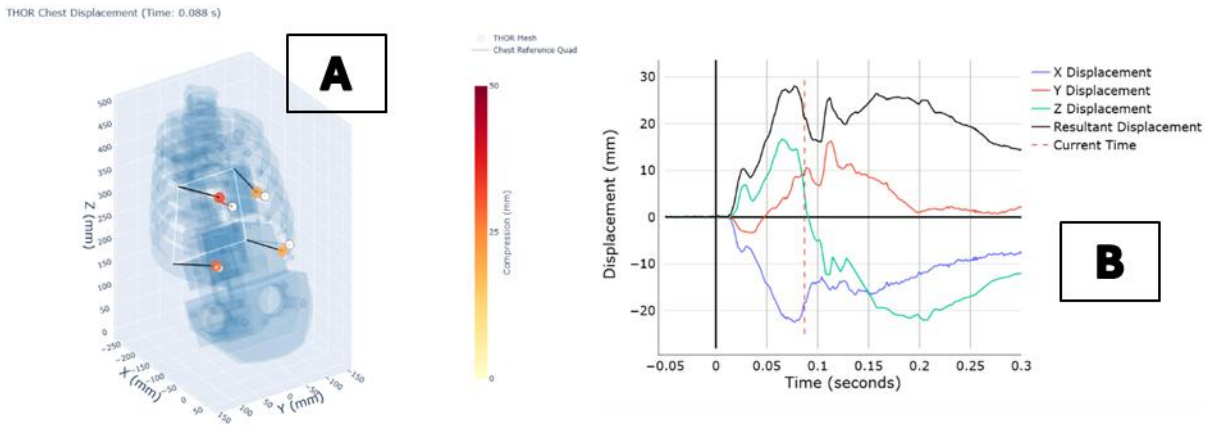

Figure 10. Sample results of the THOR 3D Deflection tool including (A) three-dimensional animation of IR-TRACC deflection and (B) time histories of IR-TRACC displacement.

Vehicle metrics: The final section of the left-side panel is the Vehicle Analysis tools. This includes the NCAP star rating calculation, load cell analysis, toe pan intrusion analysis, and pedestrian hood impact analysis tools.
The NCAP star ratings tool was built to replicate the injury metrics and calculation methods used by the official U.S. NCAP star rating system. Users can input CTD test numbers associated with the frontal, side moving deformable barrier (MDB), and side pole tests for a specific vehicle model, and the relative risk scores and star rating for each

occupant in each test will be automatically calculated. An example output of the NCAP star rating output interface is shown in Figure 11.

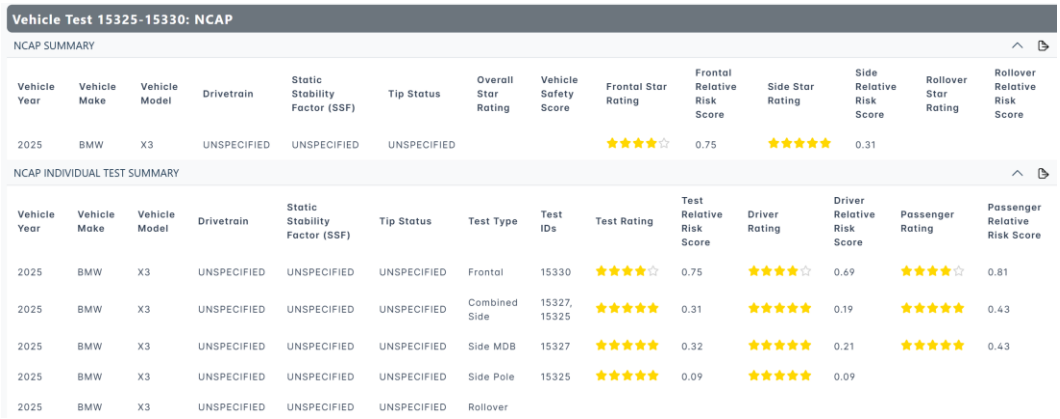

Figure 11. Sample NCAP rating results generated by WebSAT.

The load cell wall analysis tool enables the user to visualize the distribution of force and impulse applied to the barrier during a frontal impact test. This module provides insight into how the vehicle structure transfers load to the wall throughout the crash event, including parameters such as vehicle displacement, vehicle velocity, total barrier force, force by row, force by column, height of force (HOF), and initial stiffness. Different barrier configurations can be displayed based on the specific setup used in the crash test. In the heat map of maximum forces on the load cell wall, the user can overlay the vehicle outline, display the average height of force (AHOF), and optionally invert the X-axis view, as shown in Figure 12.

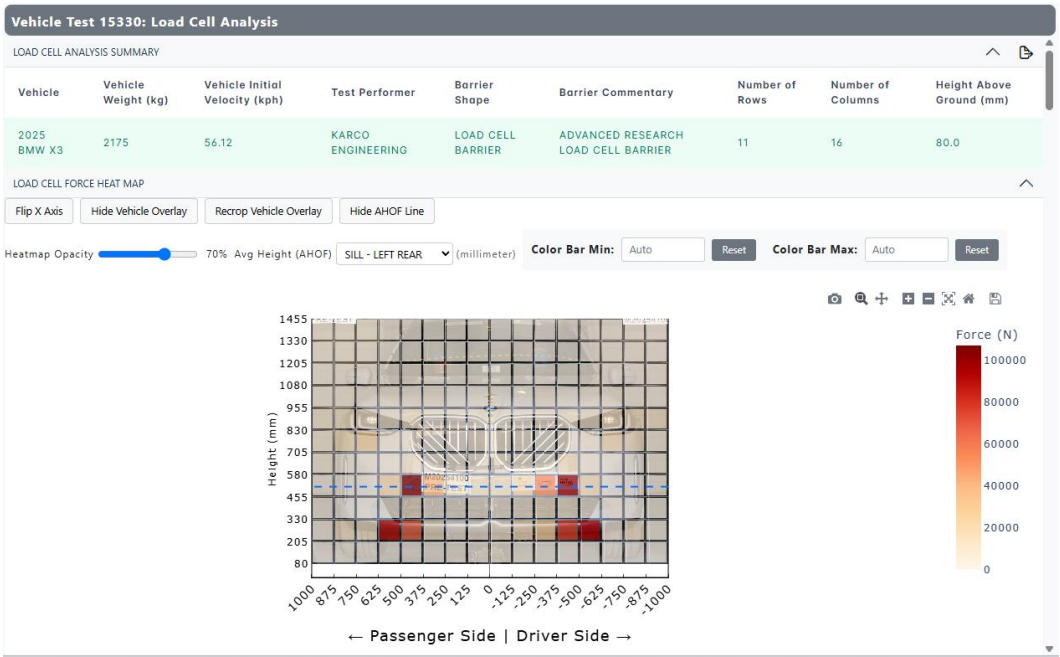

Figure 12. Sample load cell analysis results generated by WebSAT.

A vehicle intrusion analysis tool (Figure 13) is available to identify potentially injurious levels of toe-pan intrusion during frontal crash events. The tool outputs a pair of pre-crash and post-crash geometry overlays representing the contour of the toe pan. Additionally, a bar chart reports the magnitude of intrusion at key positions on the toe pan with reference to four thresholds per measurement location: good (Green), acceptable (Yellow), marginal (Orange), poor (Red).

Figure 13. Sample toe pan intrusion results generated by WebSAT.

The pedestrian hood area function generates a bird's-eye mockup of a vehicle's hood, highlighting key areas such as the HIC Unlimited Area, Child Headform Test Area, Adult Headform Test Area, Hood Top, Hood Area, and Test Area. Wrap Around Distance (WAD) lines are displayed on the hood top to define boundaries for these regions. Impact points indicate where a child or adult headform contacted the hood during pedestrian head impact tests proposed for FMVSS No. 228 Pedestrian Head Protection. Figure 14 shows an example of the Pedestrian Hood function for a sample dataset created for development purposes from the CTD.

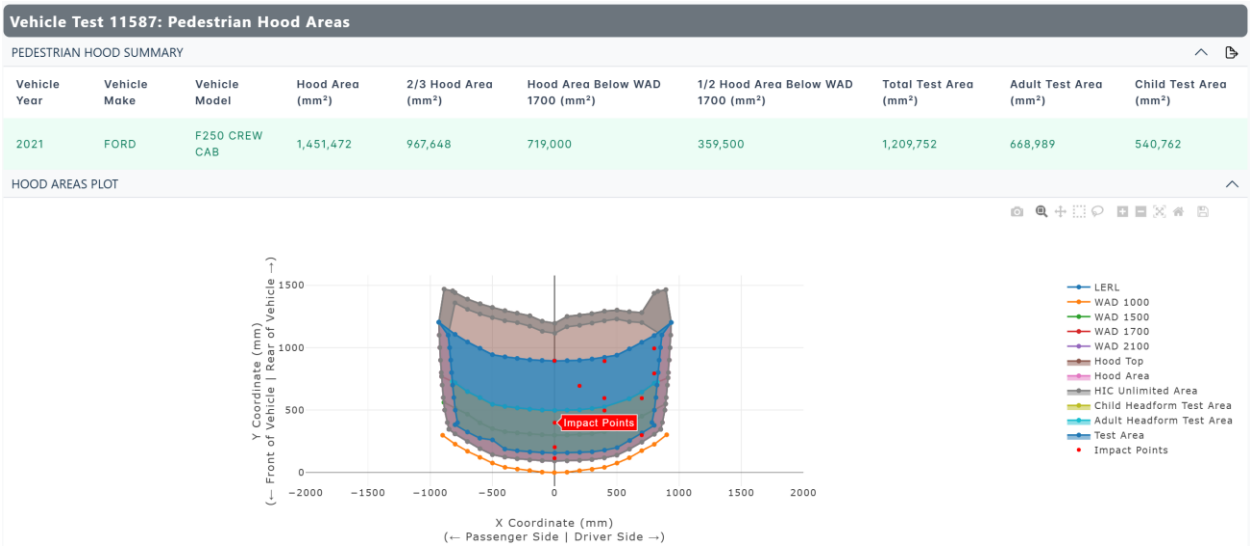

Figure 14. Sample pedestrian hood analysis results generated by WebSAT.

WebSAT performance and reliability: While WebSAT has had active staging and production deployment for more than a year, effort has been focused on implementing NHTSA-requested functionality and improving ease of use. The application has not yet been made available to the general public. As such, no production-scale usage metrics are available yet. However, internal load testing indicates stable behavior under moderate concurrent load of up to 100 requests/second--including the computationally expensive injury metric calculations--in local development environments. Further scaling of load tests in development environments has been precluded by CPU usage of the load testing software itself.

WebSAT's architecture supports horizontal scaling by adding additional Celery worker nodes to handle a greater number of computationally intensive tasks concurrently. As the application's state model consists only of transient, re-computable data, any runtime component can be updated or redeployed with only minimal impact to end users. The demonstration deployment on AWS uses CloudWatch and ECS monitoring to ensure availability of the application and timely notification of system administrators in the event of an outage.

DISCUSSION

The modernization of the NHTSA CTD and the development of WebSAT collectively represent a significant advancement in how vehicle crash test data are managed, processed, and analyzed within NHTSA's ecosystem. This paper described the four-phase modernization of the CTD—migrating to a cloud environment, introducing a secure data upload portal, adding the CADB, and expanding existing schemas for pedestrian crashworthiness—and detailed the design and validation of WebSAT, a unified, browser-based signal analysis platform. Together, these efforts have modernized both the data infrastructure and analytical tools that support federal vehicle safety research, regulation, and consumer information programs.

The updates to the CTD have provided many benefits. First, the CTD is now compliant with all DOT-wide IT and cybersecurity guidelines. Additionally, by migrating to cloud services, NHTSA is saving a considerable amount in annual costs that can be used for other key IT priorities. By streamlining the data submission and review process through a unified web portal, NHTSA is able to save time, reduce errors, and lower the burden to test labs resulting in lower test costs. Each individual user is no longer required to download a set of executable files to their local machine, and NHTSA is no longer required to maintain and update these executable files to keep pace with consumer operating systems. The new APIs and online search and test information pages provide the public with better access to NHTSA research, NCAP, and compliance test data. The updates from "version 5" to "version 6" and the development of the new CADB provide NHTSA the ability to store more test information than before for a wider variety of test types that will serve the agency and the motor vehicle safety community for years to come. The use of best practices and documentation of the modernized CTD will also ease future DME and O&M needs, reducing costs.

An additional benefit of the modernized CTD is the ability for researchers to access the information in the database programmatically using the API. An example of API access using the Python programming language is shown in Appendix D (Figure D1). This example queries the vehicle database to determine the test numbers, vehicle model year, make, and model, and test weight for all frontal NCAP and Optional NCAP tests over a given range of model years. This example can be expanded to extract further information from these tests using additional API end points, including instrumentation information, occupant information, test reports, test photos, and test videos. Additionally, the Ratings API [32] can be queried in a similar fashion to determine rating information associated with these tests.

The WebSAT platform provides several major benefits for engineers, researchers, and the general public. First, it eliminates the need for maintaining and distributing dozens of standalone desktop programs, simplifying maintenance and ensuring that all users access the same validated algorithms through a single web interface. Second, WebSAT directly interfaces with the CTD via an API, enabling seamless data retrieval and reducing errors associated with manual downloads and file management. Third, because WebSAT's backend (WebSAT-Signal) implements validated algorithms consistent with established SAE practices and federal safety standards, the analytical results it generates are accurate and reproducible. The platform's continuous integration and automated testing workflows ensure that future updates maintain numerical consistency with historical calculations. Additionally, WebSAT enhances transparency and accessibility by providing centralized documentation and version-controlled computation libraries, ensuring that injury metric definitions and processing logic are publicly

reviewable. The system's architecture allows for modular expansion, enabling the future integration of additional analysis modules (e.g., crash avoidance, pedestrian protection) as testing programs evolve.

LIMITATIONS

Although the modernization of the CTD and development of WebSAT represent major advancements in NHTSA's data and analysis infrastructure, several limitations should be noted. First are the limitations imposed by DOT-wide IT and cybersecurity guidelines restricting total developmental freedom. At the time of writing this paper, updates from "version 5" to "version 6" of BioDB and VehDB have not yet been completed but are expected to be rolled out to the public in 2026. Despite the considerable resources used in the DME of CTD, there will still be the need for O&M resources in perpetuity to keep technologies up to date and secure as well as assisting with user help desk tickets. Data that predates these CTD enhancements has had limited updates. Most of the new tables and fields introduced in version 6 schemas will be blank for older tests. Furthermore, crash avoidance data that predates the creation of CADB has not been populated.

With respect to WebSAT, several limitations reflect its status as a first-generation web-based analytical platform. Because WebSAT operates within a browser–server model, an active network connection is required for all analyses, and offline processing is currently unsupported. The platform is also restricted to data housed within the CTD and does not yet permit the upload or processing of user-supplied datasets. This design decision preserves data integrity and ensures that all analyses are traceable to federally curated sources but limits use for proprietary or pre-publication research. As a cloud-based application, WebSAT's performance is influenced by server capacity and concurrent user activity, which may affect responsiveness during computationally intensive or large batch analyses. Additionally, while the WebSAT-Signal library maintains comprehensive test coverage and has been validated against legacy desktop applications, automated testing for the WebSAT-App repository—particularly the front-end components—remains limited and is an area for future improvement.

Finally, WebSAT is not yet optimized for mobile devices and has not been integrated with the CADB schema for crash avoidance analysis. NHTSA has also requested new and expanded functionality that has not yet been developed. These enhancements, along with improved caching efficiency, performance telemetry, and expanded automated testing, are planned for future development phases.

CONCLUSIONS

The modernization of NHTSA's CTD and the development of the WebSAT platform collectively deliver a secure and sustainable data ecosystem for crash test analysis. The transition to cloud-based infrastructure has increased efficiency, reduced costs, and improved accessibility for internal and external users. The addition of new schema such as the CADB and expanded ComDB capabilities ensures that the CTD can accommodate emerging vehicle safety technologies and pedestrian protection testing.

WebSAT, built upon validated and standardized computational methods, provides an integrated, browser-based environment for performing crashworthiness analyses consistent with federal regulations and industry standards. Its API-driven architecture and intuitive user interface enable efficient evaluation of injury metrics and vehicle performance without reliance on legacy desktop applications.

The new CTD is publicly available and accessible at https://www.nhtsa.gov/research-data/research-testing-databases/. The WebSAT platform is currently not publicly available but will eventually be accessible at https://websat.nhtsa.dot.gov.

DISCLAIMER

The authors have no conflicts of interest to disclose.

ACKNOWLEDGMENTS

Funding for the development of the WebSAT platform was provided by National Highway Traffic Safety Administration IDIQ task order 693JJ923R000183. The authors would like to thank Steve Summers for his expertise and leadership in modernizing the NHTSA signal analysis tools and crash test database.

REFERENCES

1. United States Congress. National Traffic and Motor Vehicle Safety Act of 1966. 15 U.S.C. § 1381 1966.
2. Gepner B, Bollapragada V, Acosta SM, Park G, Forman J. Comparison of THOR LX Xversion and Dorsiflexion Response in Component Tests, Sled Tests and Full Vehicle Crash Tests. In: 25th International Technical Conference on the Enhanced Safety of Vehicles (ESV). Detroit, Michigan; 2017.
3. Hallman J, Buck J, Ham SJ. Truck and Sport Utility Vehicle Front End Stiffness Corridors. SAE Technical Papers [Internet]. 2018 Apr 3 [cited 2025 Oct 21];2018-April. Available from: https://saemobilus.sae.org/papers/truck-sport-utility-vehicle-front-end-stiffnesscorridors-2018-01-0518
4. Brewer J, Patel S, Summers S, Prasad A, Mohan P. ACCURACY OF AHOF400 WITH A MOMENT-MEASURING LOAD CELL BARRIER. In: 22nd Enhanced Safety of Vehicles Conference (ESV 2011). Washington, DC; 2011.
5. National Highway Traffic Safety Administration. NHTSA Research Testing Databases [Internet]. [cited 2025 Oct 21]. Available from: https://www.nhtsa.gov/research-data/research-testing-databases/
6. National Highway Traffic Safety Administration. Component & NHTSA Database Code Library APIs [Internet]. [cited 2025 Oct 21]. Available from: https://nrd.api.nhtsa.dot.gov/swagger-ui/index.html
7. National Highway Traffic Safety Administration. Vehicle API [Internet]. [cited 2025 Oct 21]. Available from: https://nrd.api.nhtsa.dot.gov/nhtsa/vehicle/swagger-ui/index.html
8. National Highway Traffic Safety Administration. Biomechanics API [Internet]. [cited 2025 Oct 21]. Available from: https://nrd.api.nhtsa.dot.gov/nhtsa/biomechanics/swagger-ui/index.html
9. National Highway Traffic Safety Administration. Crash Avoidance Database Library APIs [Internet]. [cited 2025 Oct 21]. Available from: https://nrd.api.nhtsa.dot.gov/nhtsa/cadb/swagger-ui/index.html
10. National Highway Traffic Safety Administration. 89 FR 76922 - Federal Motor Vehicle Safety Standards; Pedestrian Head Protection, Global Technical Regulation No. 9; Incorporation by Reference [Internet]. Vol. 89. Federal Register; 2024 [cited 2025 Oct 21]. p. 76922–7010. Available from: https://www.regulations.gov/document/NHTSA-2024-0057-0001
11. National Highway Traffic Safety Administration. 89 FR 93000 - New Car Assessment Program: Crashworthiness Pedestrian Protection [Internet]. Vol. 89. Federal Register; 2024 [cited 2025 Oct 21]. p. 93000–36. Available from: https://www.regulations.gov/document/NHTSA-2024-0078-0001
12. Django Software Foundation. Django [Internet]. Available from: https://www.djangoproject.com/
13. You E. Vue.js [Internet]. [cited 2025 Oct 22]. Available from: https://vuejs.org/
14. Meta Platforms Inc. Docusaurus [Internet]. [cited 2025 Oct 22]. Available from: https://docusaurus.io/
15. Evans D. WhiteNoise [Internet]. [cited 2025 Oct 22]. Available from: https://whitenoise.readthedocs.io
16. LF Projects L. OpenTofu [Internet]. [cited 2025 Oct 22]. Available from: https://opentofu.org/
17. Amazon Web Services Inc. Amazon Web Services [Internet]. [cited 2025 Oct 22]. Available from: https://aws.amazon.com/
18. Ask Solem. Celery [Internet]. [cited 2025 Oct 22]. Available from: https://docs.celeryq.dev/en/stable/
19. Redis Ltd. Redis [Internet]. [cited 2025 Oct 22]. Available from: https://redis.io/
20. PrimeTek Informatics. PrimeVue. [cited 2025 Oct 22]; Available from: https://primevue.org/
21. Wallace E. esbuild [Internet]. [cited 2025 Oct 22]. Available from: https://esbuild.github.io/
22. Yarn [Internet]. [cited 2025 Oct 22]. Available from: https://yarnpkg.com/
23. OpenJS Foundation. Jest [Internet]. [cited 2025 Oct 22]. Available from: https://jestjs.io/
24. Encode OSS Ltd. Django REST Framework [Internet]. [cited 2025 Oct 22]. Available from: https://www.django-rest-framework.org/
25. Franzel T. drf-spectacular [Internet]. [cited 2025 Oct 22]. Available from: https://drf-spectacular.readthedocs.io/en/latest/
26. SmartBear Software. Swagger UI [Internet]. [cited 2025 Oct 22]. Available from: https://swagger.io/tools/swagger-ui/
27. Redocly Inc. Redoc [Internet]. [cited 2025 Oct 22]. Available from: https://github.com/Redocly/redoc

28. Django Software Foundation. Daphne [Internet]. [cited 2025 Oct 22]. Available from: https://github.com/django/daphne
29. National Highway Traffic Safety Administration. Product Information Catalog and Vehicle Listing.
30. National Highway Traffic Safety Administration. CTD - Crash Avoidance JSON Schema [Internet]. [cited 2025 Oct 21]. Available from: https://nrd-static.nhtsa.dot.gov/crashAvoidanceSchema.json
31. National Highway Traffic Safety Administration. 89 FR 95916 - New Car Assessment Program Final Decision Notice-Advanced Driver Assistance Systems and Roadmap [Internet]. Vol. 89. Federal Register; 2024 [cited 2025 Oct 21]. p. 95916–6083. Available from: https://www.regulations.gov/document/NHTSA-2024-0077-0001
32. National Highway Traffic Safety Administration. NHTSA Datasets and APIs [Internet]. [cited 2025 Oct 21]. Available from: https://www.nhtsa.gov/nhtsa-datasets-and-apis

APPENDIX A

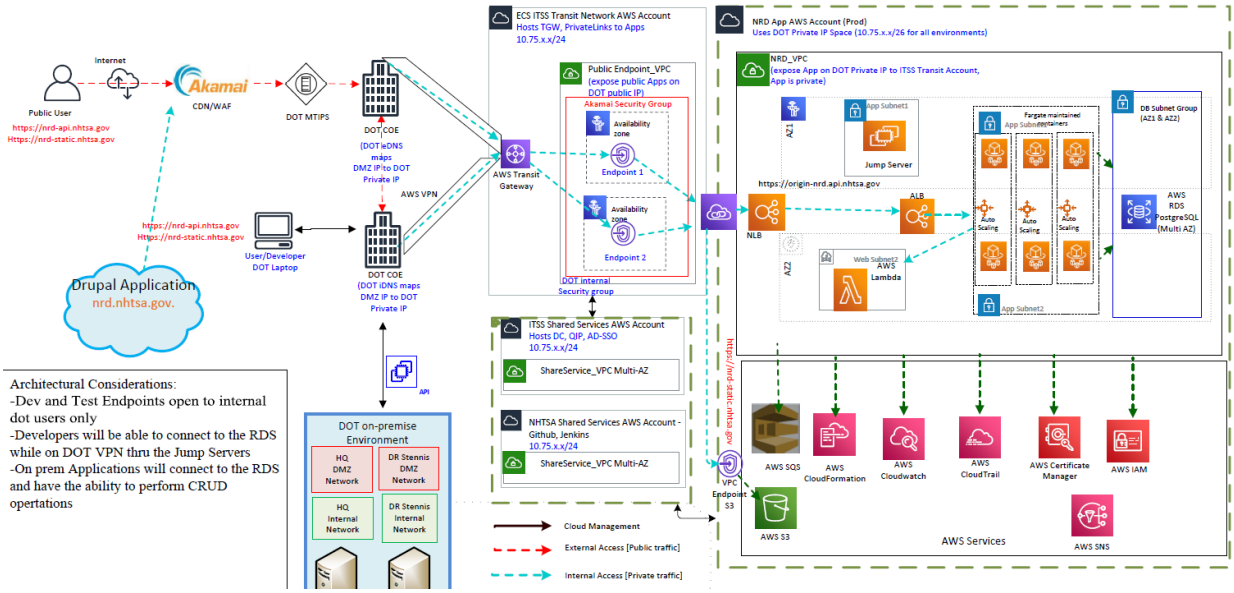

Figure A1. Technical architecture of the modernized CTD.

Infrastructure:

- AWS: S3, Fargate, ECS, ECR
- Database: RDS PostgreSQL
- CI/CD: Jenkins/IAC (Infrastructure as Code), Terraform
- Akamai

API:

- Spring Boot REST API
- Maven

User Interface:

- vue.js
- Drupal 8

Other technologies:
Cypress, JaCoCo, SonarQube, JMeter, New Relic, Docker, AWS Lambda

APPENDIX B

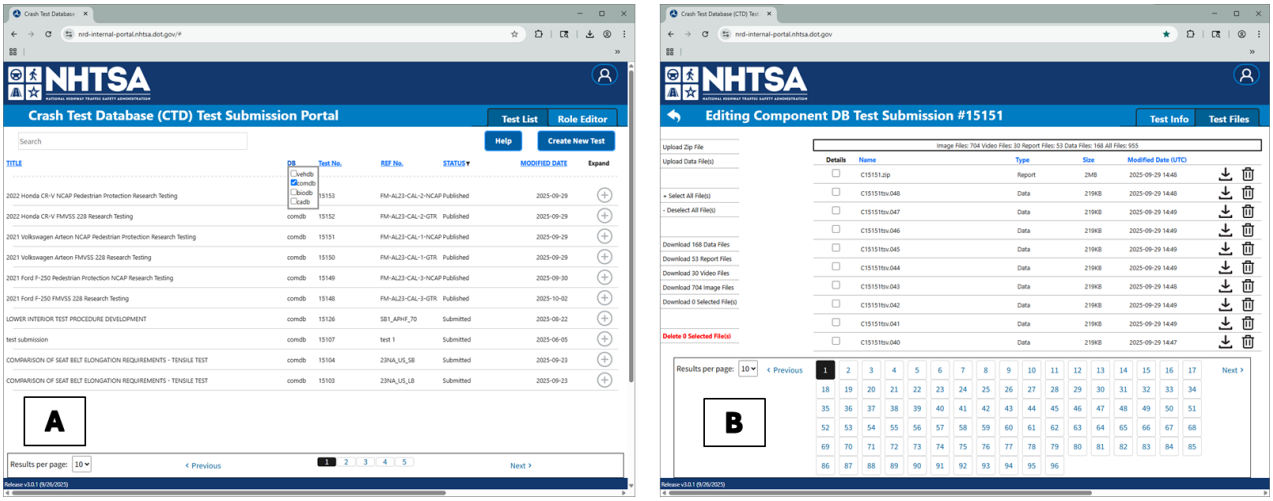

Figure B1. The modernized administrative interface (left) and media upload interface (right) of the CTD portal.

APPENDIX C

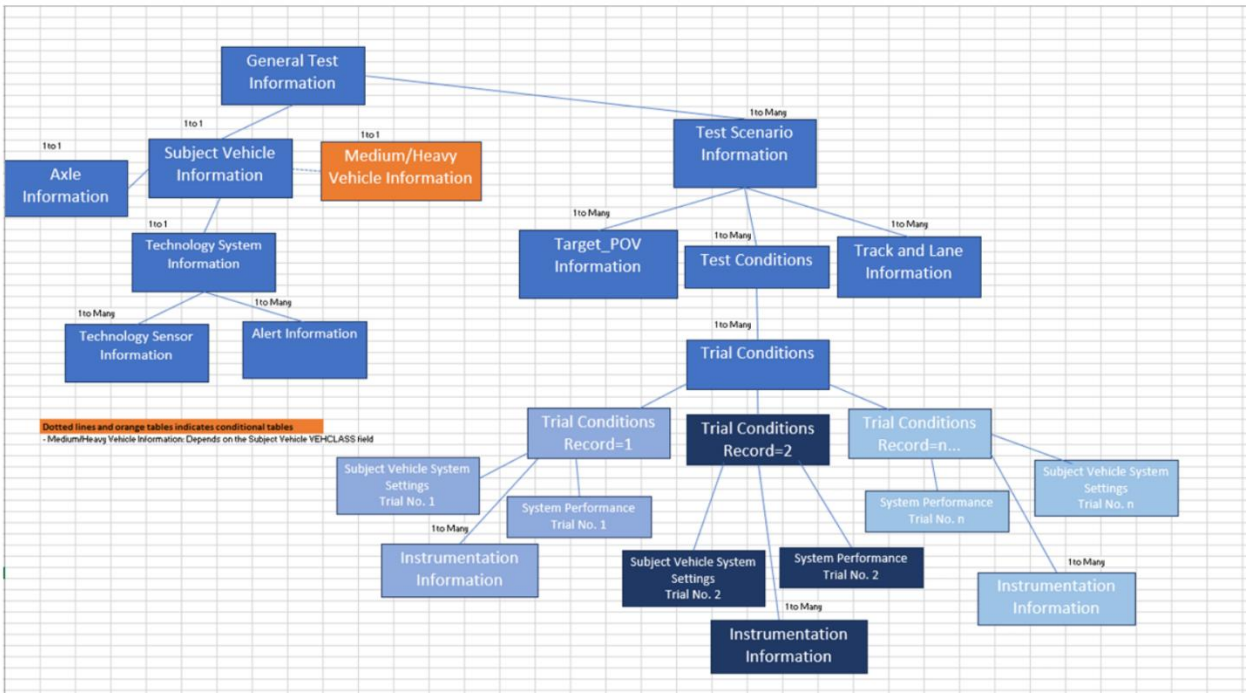

Figure C1. Relational diagram for the CADB.

APPENDIX D

```python
import requests

def getFrontalNCAPtests(MYstart, MYend):
    # find frontal NCAP crash tests
    testlist = []
    for TSTTYP in ['NCA','ONC']: # New Car Assessment Test; Optional NCAP
        TSTCNF = 'VTB'  # vehicle into barrier
        IMPANG = 0  # full frontal
        CLSSPD = 56 # km/h
        startYEAR = MYstart
        endYEAR = MYend
        reqURL = (f"https://nrd.api.nhtsa.dot.gov/nhtsa/vehicle/api/v1/"
            f"vehicle-database-test-results/by-search?"
            f"closingSpeedFrom={CLSSPD - 2}&"
            f"closingSpeedTo={CLSSPD + 2}&"
            f"count=99999&"
            f"impactAngleFrom={IMPANG}&"
            f"impactAngleTo={IMPANG}&"
            f"orderBy=testNo&"
            f"sortBy=DESC&"
            f"testConfiguration={TSTCNF}&"
            f"testType={TSTTYP}&"
            f"modelYearFrom={startYEAR}&"
            f"modelYearTo={endYEAR}")
        response = requests.get(reqURL)
        matchingTests = response.json()['results']

        for thisTest in matchingTests:
            testlist.append([thisTest['testNo'],
                             thisTest['modelYear'],
                             thisTest['vehicleMake'],
                             thisTest['vehicleModel'],
                             thisTest['vehicleTestWeight']])

    testlist.sort(key=lambda x: x[0])
    # prepend header
    testlist.insert(0,['TSTNO','Year','Make','Model','Test Weight (kg)'])
    return testlist

getFrontalNCAPtests(MYstart=2024, MYend=2025)
```

Figure D1. Python script demonstrating API access to the NHTSA CTD.