Defence Research and Development Canada
Recherche et développement pour la défense Canada

CAN UNCLASSIFIED

Sharik 3.0 Design Proposal

Alexei Zenin

Matthew Lakier
Michael Chen
Minna Wu
University of Toronto

Prepared by:
University of Toronto
27 King's College Cir, Toronto, ON M5S 3H7

Technical Authority: Shadi Ghajar-Khosravi
Contractor's date of publication: November 2017

Defence Research and Development Canada

Contract Report
DRDC-RDDC-2018-C212
November 2018

CAN UNCLASSIFIED

Canada

IMPORTANT INFORMATIVE STATEMENTS

This document was reviewed for Controlled Goods by Defence Research and Development Canada using the Schedule to the *Defence Production Act*.

Table of Contents

Executive Summary

This design proposal addresses the improvement of Sharik – a web-based intelligence sharing tool that is employed by the client DRDC. The client has requested a redesign of Sharik to address scalability and usability issues.

The Requirements section of this document details the framing of the design problem in terms of the problem statement, stakeholders, functions, objectives, constraints, and service environment. This section was formulated based on client consultation, research into standardized practices, and the design team's prior experience.

The Design Alternatives section details the design iteration process. It deconstructs the design problem into high-level architectural views, low-level software component views, and the UI design view. Each iteration process began by producing solution alternatives, which were either surveyed from existing designs or produced from heuristics, which was followed by the elimination of unviable alternatives based on requirements highlighted in the Requirements section. Once the design team exhausted the solution alternative space following one or many iterations, either one design alternative was chosen or if necessary, riskier design decisions were left open in expectation of new information later in the implementation phase that would inform a better decision. The implementation of the chosen alternatives is outlined at a high-level in the Project Management Plan section.

The engineering team is planning to move forward with developing a web application, accessible by browser. Sharik will be structured in three distinct components: client-side, server-side, and database. A single page application architecture was chosen to be used with technologies such as Vue.js and Java's Spring framework.

Three core features were identified as data-entry, proposition visualization within a concept map, and collaborative slide editing which will be implemented in time for the client's focus group during early March 2018. This will evaluate the viability of the tool the design team builds. The Project Management Plan and constraints in the Requirements section reflect the prioritization of the core features. Certain constraints, deemed not essential to the scope of this project's focus group deadline, were moved out of the constraints section to make sure the team can meet the requirements of the client on time.

1 Requirements

1.1 Problem Statement

DRDC (the client) has built a web-based intelligence sharing tool called Sharik. It aims to provide the ability for intelligence analysts to collaborate on intelligence gathering and sharing.

The client has asked the engineering team to redesign the web based tool from the ground up to address scalability and usability issues. This means the engineering team needs to design a software solution which includes the data models, architecture, back-end processes and user interfaces for analysts to interact with. The client has also made clear that the solution must be able to serve multiple users distributed geographically across the world (i.e. over a network). The client will host a focus group for the solution's intended users in mid-March, for which the team is required to deliver several fully functional core features as outlined in the constraints.

1.2 Background

The Canadian Armed Forces (CAF) need tools to support their intelligence analysts who deal with many various sources and types of information at scale. Collators or collection assets are the personnel who forward such information to the analysts who then process and produce shareable intelligence [1].

Intelligence analysis has become bottlenecks for other countries such as EU members, and the UK [2]. Intelligence analysis tools like Sharik can help make the analyst's tasks more efficient, thus making the intelligence lifecycle (IC) process more streamlined.

There are 4 stages to the intelligence lifecycle: *Direction, Collection, Processing,* and *Dissemination*. During the *Direction* stage, the Commanding Officer (CO) of the operation provides the Commander's Critical Information Requirements (CCIRs), which are then assigned to a group of information officers and their subordinates. This group breaks down each CCIR into many Priority Intelligence Requirements (PIRs) which are all part of an Intelligence Collection Plan (ICP). The *Collection* stage involves breaking down the PIRs into Information Requirements (IRs) for analysts to work on. The analyst will then forward this IR to a collator or make a Request For Information (RFI) to an external department. During the *Processing* stage, analysts combine the results of the IRs to answer the PIR. Finally, once a PIR or a set of PIRs have been answered, the results go through the *Dissemination* stage, in which the intelligence is shared through some medium (e.g., verbally, files). The structure of the ICP is depicted below in Figure 1. Note from the bottom box of Figure 1 that the Sharik project space provides the ability to add notes, wikis, propositions (relationships), and other data for intelligence items that are a part of the ICP (see Appendix A for detailed definitions) [1].

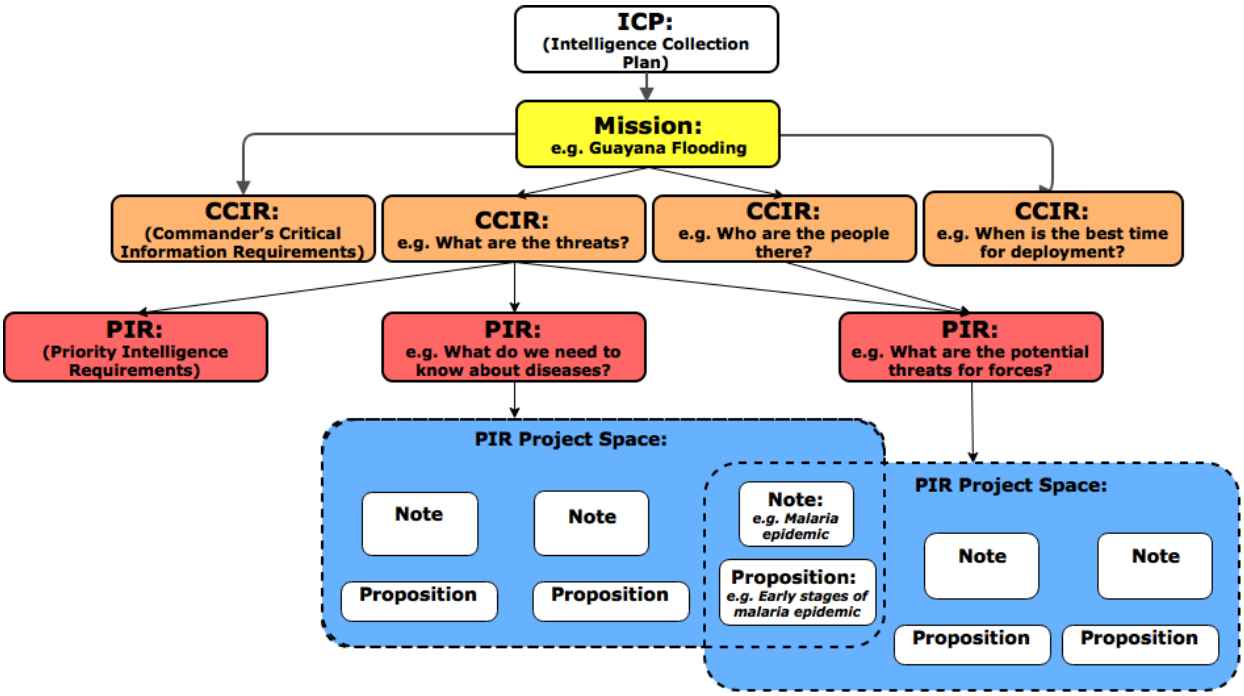


Figure 1. The hierarchical decomposition of IC elements in Sharik. The Sharik tool allowed users to upload an ICP which breaks down to the corresponding sub-elements: Mission, CCIR, PIR. In this case the above intelligence elements depict an intelligence mission revolving around the hypothetical flooding in Guayana. As can be seen the intelligence mission is then decomposed into various intelligence sub-elements which help accomplish the mission. The intelligence items on the far left are used as defining elements, where the rest of the elements address the mission in some way. Sharik's previous version did not function in the exact way as shown above (a revision to how the ICP is structured has made the previous version of Sharik outdated). In this version of the ICP, PIRs can be part of many CCIRs (the far right PIR is part of two CCIRs), along with notes and propositions having the ability to be part of multiple PIRs as well (depicted by the shared pool in between the project spaces).

Based on conversations with the client, Sharik has focused on the latter three stages of the intelligence lifecycle by providing analysts with data entry components, relationship visualizations between intelligence items (Concept Map feature or CMap), and presentation slide generation to share intelligence.

The client would like to address a list of usability and responsiveness issues found through a usability study of the initial iteration of Sharik involving five intelligence analysts. These include bugs in the UI, backend processing taking too long (2–3 seconds) along with scalability, and design issues.

1.3 Stakeholders

The design should also take into consideration the needs and requirements from the following stakeholders.

- **Intelligence analysts:** Intelligence analysts will use Sharik to assist with sensemaking. The tool will allow them to analyze information and prepare it for dissemination in less time. The tool should maximize the data-entry speed, minimize presentation creation time of analysts, and allow collaboration between analysts to make sense of large volumes of information.
- **Collectors:** Collectors provide information to analysts, and may do so using Sharik. The tool should provide a means for collectors to share the information they have collected with intelligence analysts.
- **Commander(s):** Commanders will create ICPs which are ingested by the tool and decomposed into CCIRs and PIRs. They will receive intelligence results from analysts in the form of several slides. The tool should assist analysts in creating slides that are clear and relevant to the commanders.
- **DRDC (including Shadi Ghajar-Khosravi and Peter Kwantes)**: DRDC would like to reduce the amount of time intelligence analysts spend searching for information and therefore maximize the amount of time that can be dedicated to analysis. Also, DRDC would like the intelligence information to remain secure by retaining all data on DRDC servers. They will communicate with the team to ensure that the tool addresses the needs of the intelligence analysts. The tool will be deployed to help achieve these goals.
- **Future code maintainers:** The created tool will need to be maintained and extended in future iterations. The implementation of the tool should be understandable and extensible for future code maintainers, using practices such as code documentation and unit tests.
- **Capstone team:** The Capstone team will design and prototype the tool, communicating regularly with the client to ensure that requirements properly reflect the needs of the analysts, and that tool prototypes satisfy the requirements. The team will use its collective, multi-disciplinary (electrical and computer engineering and industrial engineering) knowledge to produce an optimal solution given the project timeline.

1.4 Functions

The functions of this tool should allow multiple analysts to access, input, analyze, and transform information required for the intelligence cycle. Specifically, it should:

- Allow collaboration of analysts through the Sharik environment
- Create storage and retrieval mechanisms for intelligence information
- Enable analysts to enter data as notes and propositions
- Transform information into easy-to-understand visual concept maps
- Allow analysts to access tools that support the dissemination process such as collaboratively-built slideshow presentations

1.5 Objectives

The project is guided by the following list of objectives and their corresponding secondary objectives to maximize design quality and user satisfaction. These objectives were chosen after analysis of client need and usability feedback data, and then matching them to appropriate design for excellence criteria, metrics, and definitions in ISO/IEC 25010:2011 and ISO/IEC 25023:2016 – international standards for evaluating software quality [3] [4].

Criteria and metrics were chosen based on relevancy to the tool to be designed, and to match achievable expectations for what the team can measure in this design project. Finally, a pair-wise comparison chart was created to compare priorities between each pair of objectives (see Appendix B). The order of objectives listed below is from highest to lowest priority.

Design for Excellence (DfX)	**Sub-category for DfX**	**Metric ID**	**Target for Project**	**Measurement Function**
Performance Efficiency	Time Behavior	PTb-5-G/Mean throughput	An increase in the mean throughput of jobs completed by a factor of two relative to existing design. We define a job as a typical use case of Sharik (e.g. inputting a note).	$X = \sum_{i=1 \text{ to } n} (A_i / B_i) / n$ A_i = Number of jobs completed during the i-th observation time B_i = i-th observation time period n = Number of observations
		PTb-1-G/Mean response time	An increase in mean response time of page loads by a factor of two relative to existing design	$X = \sum_{i=1 \text{ to } n} (A_i) / n$ A_i = Time taken by the system to respond to a specific user task or system task at i-th measurement n = Number of responses measured
Usability	Learnability	ULe-1-G/User guidance completeness	100% of functionality described in user documentation and/or help facility	X = A/B A = Number of functions described in user documentation and/or help facility as required B = Number of functions implemented that are required to be documented
		ULe-2-S/Entry fields defaults	100% of applicable entry fields have placeholder data to express the format and variety of expected data	X = A/B A = Number of entry fields whose default values have been automatically filled in during operation B = Number of entry fields that could have default values

		ULe-3-S/Error messages understandability	100% of error codes state the reason of occurrence and suggest ways of resolution	X = A/B A = Number of error messages which state the reason of occurrence and suggest the ways of resolution where this is possible B = Number of error messages implemented
	Operability	UOp-2-G/Message clarity	100% of messages delivered to the user convey clear outcomes or instructions to the user	X = A/B A = Number of messages that convey the right outcome or instructions to the user B = Number of messages implemented
	User interface aesthetics	UIn-1-S/Appearance aesthetics of user interfaces	Decrease average user interface and overall design issues by a factor of two by increasing satisfying interactions with the user	X = A/B A = Number of display interfaces aesthetically pleasing to the users in appearance B = Number of display interfaces
Maintainability	Testability	MTe-1-G/Test function completeness	50% of code is covered by tests	X = A/B A = Number of test functions implemented as specified B = Number of test functions required
Reliability	Maturity	RMa-3-G/Failure rate	Typical use cases experience failures (as defined in Appendix A) 0% of the time in a staging environment. Tests will be performed each time from a fresh deployment from scratch.	X = A/B A = Number of failures detected during observation time B = Duration of observation
	Recoverability	RRe-1-G/Mean recovery time	Average time to re-initiate operation after a failure is below five minutes	$X = \sum_{i=1 \text{ to } n} A_i / n$ A_i = Total time to recover the downed soft-ware /system and re-initiate operation for each failure i n = Number of failures

Functional Suitability	Functional Completeness	FCp-1-G/Functional coverage	Maximize the proportion of the specified optional feature functions that are implemented (see Section 1.7) No goal is set as any time remaining after meeting preceding objective goals will be spent here	$X = 1 - A/B$ A = Number of functions missing B = Number of functions specified

1.6 Constraints

A list of mandatory constraints was grouped into data-entry, dissemination, business rules, and logistics. These are what the final solution must have. Three core features were discovered in client consultations which are needed for the client's early March focus group trial which are: data-entry of notes and propositions, visualizing of propositions into a CMap, and slide generation. The list below describes the core features and other needed supporting constraints.

Data-Entry:

Data-entry constraints have to do with the method in which analysts will input data into Sharik.

1. Shall have logical intelligence components represented in the tool (PIR, IR, CCIR, Mission).
2. Shall have the ability to attach files and website links to propositions and notes.
3. Shall have note and proposition metadata that keeps track of time of submission and that links content to the contributing analyst.
4. Shall provide an auditing feature which will allow users to see changes to notes or propositions.
5. Shall contain file management to allow for the creation of a common pool of files that can be shared between CCIRs; proposition and note attachments would come from this pool.

Dissemination:

Dissemination constraints have to do with the visualization and presentation of the relationships within the data.

1. Shall have the ability to place information from PIRs into a presentation slideshow.
2. Shall have a concept map feature for visualizing propositions graphically to obtain a "common intelligence picture" (i.e. a visual map of the relationships between different entities among collected information). Shall have include the ability to filter propositions by user and properties.

Business Rules:

Business rules are operations available and constraints that apply to individuals, teams and organizations. For example, a business rule could be "only individuals named Bob can switch teams" and this would be enforced in software by checking this condition whenever some individual requests a team switch. Some of the business rules are outlined below (this list is expected to change as the project progresses):

1. Only admins (not regular users) may add users to teams of missions.
2. Only admins may create, modify, or delete Missions; all users may create, modify, and delete CCIRs and PIRs.
3. Anyone can register in the Sharik system without the assistance of an admin.
4. PIRs shall be able to belong to multiple CCIRs within the same mission.
5. Notes and propositions shall be able to belong to multiple PIRs, or be unassigned. If a note or proposition is unassigned, then it shall be possible to add it to a mission for later assignment to a PIR.
6. Propositions shall be editable by all analysts, whereas notes shall only be editable by the creator.

Logistics:

Logistic constraints deal with constraints characterizing the surroundings of the solution.

1. Shall be affordable by DRDC: Third-party tools may be used for hosting of services. However, initial costs and operation costs shall total CAD$0 (ignoring electrical and staffing costs).
2. Shall be suitable for use on a laptop or desktop computer
    a. Shall display optimally for all screens with a resolution above or equal to XGA (1024x768) [5]
    b. Shall have a user interface dependent only on conventional hardware input devices (i.e., mouse/touchpad, and keyboard)
3. Shall not experience failure when the number of concurrent users is at least 4.
4. Shall not require wireless internet connection to run due to the service environment detailed in Section 1.7.
5. Shall not use third-party systems outside DRDC's environment (e.g. do not send data to Google Drive).

1.7 Optional Feature Requests

The following three requests were initially specified as constraints in the Project Requirements. In refining the project plan, the design team determined that there is a considerable amount of uncertainty with respect to the amount of time that certain features will take to implement in the original constraint section. The team relabelled the following constraints as optional to ensure that we can meet a certain set of core features in time for a focus group conducted by the client as mentioned earlier in the constraints. If time permits, these will be the features the team will tend to first.

1. Implement a commenting system to allow analysts to comment on propositions and notes
2. Implement a notification system that would notify analysts on changes to subscribed notes or propositions.
3. Implement a timeline view presented to visualize the dates when a proposition has occurred

The client would also like the following features if time permits.

1. Implement geo-map feature, including the ability to:
    a. Add locations to notes and propositions based on the MGRS (Military Grid Reference System)
    b. Display notes and propositions on a map or globe visualization
2. Implement the ability to save Requests-For-Information as PDFs
3. Implement the ability to save screenshots of concept maps generated by Sharik
4. Implement a communication system that would facilitate real-time messaging between analysts
5. Implement a timeline synchronization with the state of the concept map
6. Implement the ability to mark propositions or notes for future inclusion into a presentation

1.8 Service Environment

This section will briefly outline the traits of the physical and virtual environments the tool will operate in.

Physical/Virtual Environment:

- Indoors inside an office.
- LAN networks, absent of wireless internet connection.
- One laptop screen per analyst to run Sharik (no extra monitor).
- Possibility of cyber-thieves.
- Browsers used at DRDC are Internet Explorer version 11.

People:

- Intelligence analysts and anyone who may be with them, who may possibly be fatigued and/or stressed and may make human errors (as per conversation with client).
- Other DRDC employees or non-DRDC visitors who may be in the vicinity, outside of the user's group who may be interested in this project (as per conversation with client).

2 Design Alternatives

The following section will break down the various design alternatives that could be considered in achieving the requirements of the project. Each section will conclude with a selection or leave open certain parts of the selection process for later investigation.

The proposed solution must be able to allow communication between geographically distributed users, provide the functions as listed in Section 1.4 and meet the constraints of the project while

using the objectives for selection. This includes storing information persistently, coordinating information retrieval and input from users, and providing the ability to disseminate information through slides and concept maps. The users of the system were described to be desktop users with access to a joint LAN network.

2.1 Architecture

Before discussing the alternatives regarding specific software frameworks or languages, the possible architecture alternatives are discussed below.

To narrow down the set of architectural solutions, only 3-tier web architectures will be considered due to the design team having the most experience with them, and because they are among the most common architectures for creating an application for users over a network today [6] [7] [8]. Some definitions and explanation of a 3-Tier (sometimes referred to as N-Tier) web architecture based solution are presented below (definitions compiled from [6] [8] [9]).

- The **web server** is the machine which runs the core components of the application. It is responsible for maintaining connections with users over the internet and executing the functionality and business rules (domain logic) for the application (*Logic Tier*). It also is responsible for security and accessing other services such as data sources.
- A **layer** is a set of software components and assets which all focus on implementing one part of the domain logic of an application. For example, within the web server there could be a layer (e.g. several code files) dedicated to handling the network connections that users make with the server. These layers interact with each other with well-defined interfaces (software contracts on how to communicate between layers).
- A **tier** is a machine or set of machines which hosts one or more layers. For instance, the components above could all be hosted on the same tier, having all their layers run on one machine in different processes. A Tier can be thought of as being a physical separation between software components which aim to solve one set of tasks for the application/system.
- The **client** is a user machine, in this case, the machine of an intelligence analyst, accessing a web page by connecting with the web server (a different tier). This client runs a browser (like Internet Explorer), which renders and displays the user interface for the client to interact with (*Presentation Tier*). The term client usually refers to the browser program itself when in the context of software components interacting with each other. Most modern browsers support HTML, CSS and JavaScript, the staples of web interface development.
- **Data sources** are responsible for storing and retrieving data efficiently for an application (Data Tier). These are usually hosted on a separate machine from the web server.
- A **service** is an exposed layer (e.g. over some protocol over a network) which handles performing an action or returning data to a calling application. Users normally do not interact with a service but rather with a user interface which could be using a service behind the scenes.

- The **front-end** is any component related to the user interface of the application (e.g. UI component layout, logic for handling button clicks, hiding certain components from view).
- The **back-end** is any component related to process running on a machine that runs operations related to manipulating data, connecting users, and performing other integration processes between machines. Its main goal is usually to perform the required actions initiated by a user.
- A **framework** is a set of software tools used in developing certain parts of an application. It provides a software structure to accomplish the goals of the developer and usually reduces the amount of work a developer would need to do to accomplish their goal without the framework. An example would be a user interface framework like on Windows. Standardized components such as buttons, drop downs, and forms are already provided to the developer (and need not be written from scratch).

Below is a diagram which should help visualize the main components of the 3-Tier web architecture. The approach is used in modern companies today and reinforced by state-of-the-art literature as well [6] [8].

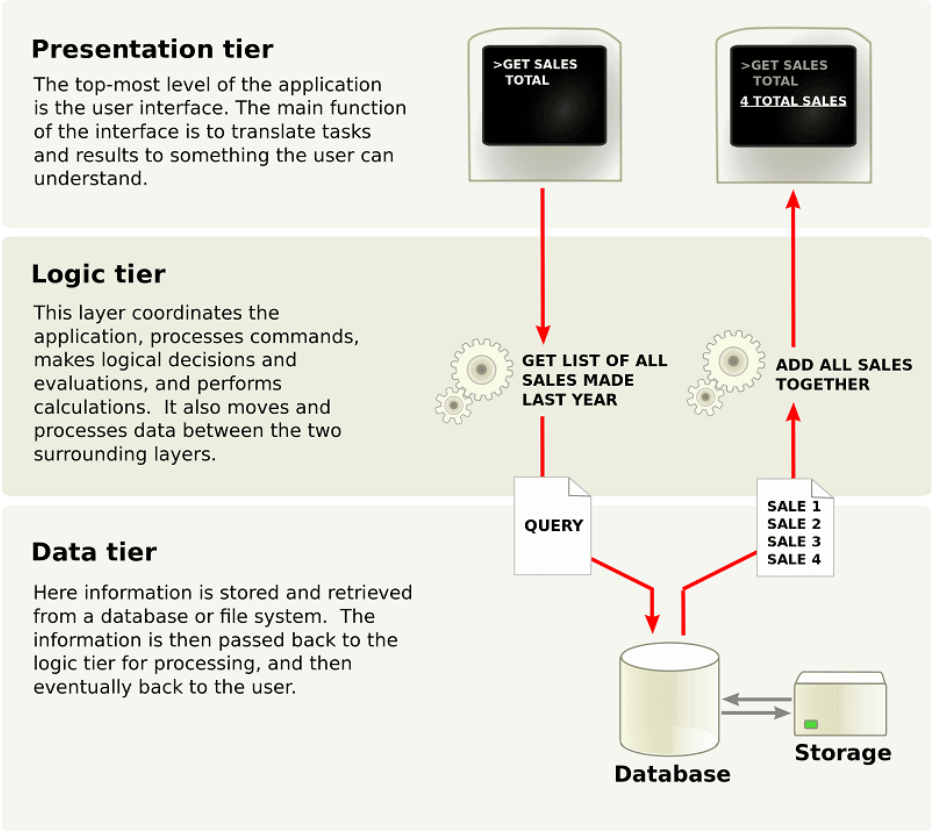


Figure 2. The three tiers displayed here correlate from top to bottom as the client, web server, and data source tiers in a 3-tier web architecture. As can be seen the Presentation tier in this diagram is a terminal requesting all the sales made last year. The Logic tier handles the processing of this request and queries the Data tier. It then handles returning the response to the Presentation tier for it to then be displayed. Taken from [8].

Given this general architecture choice there exist two main alternatives. These alternatives stem from two well-known methodologies to designing a web based application. The first is a multi-page application and the other is a single page application [10]. The concept of how each differs is illustrated below.

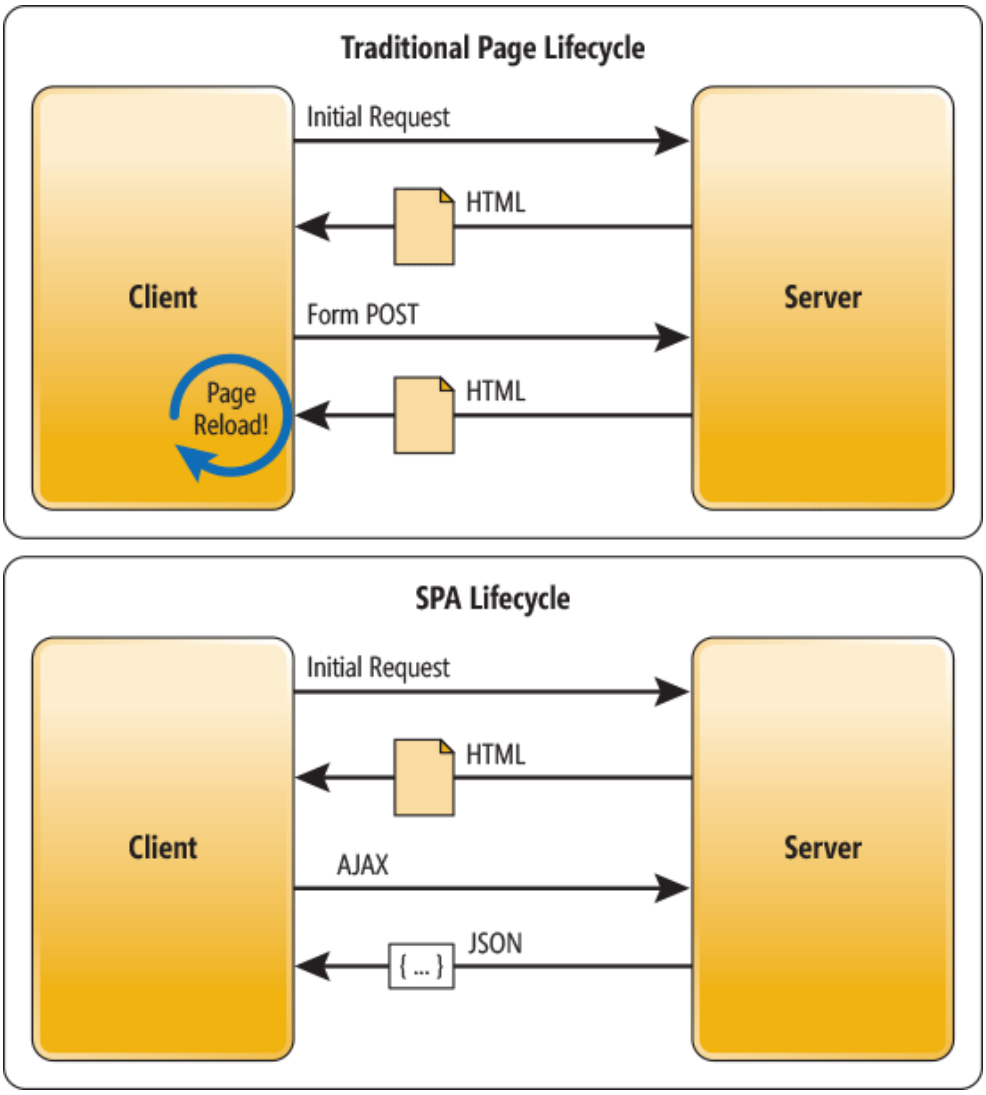


Figure 3 A contrast between the two alternatives for web based applications. Note how in the traditional or multi-page lifecycle each user action over HTTP makes the server return a new HTML page (UI). With a single page application (SPA) only the first request is needed to load the client-side script to then perform all Presentation tier functions. Subsequent calls to the server only request data through the AJAX protocol returning data formatted in the JSON standard which the Presentation tier integrates into the UI. The Data tier is omitted from the above diagram for brevity. Taken from [10].

The full look end-to-end view of both above alternatives is shown below in two figures. These figures map out each tier and some of the possible layers that one could expect to see in each architecture.

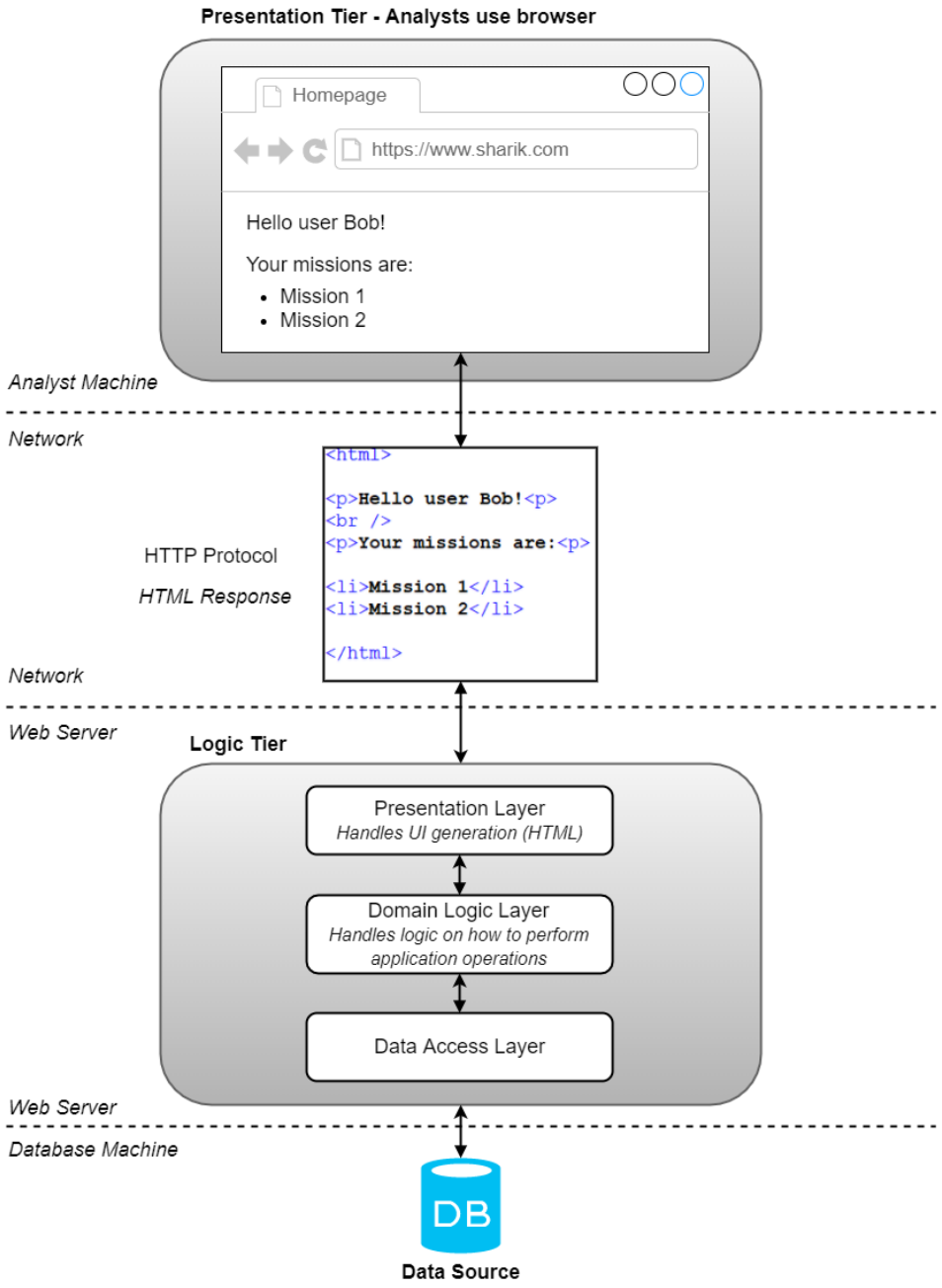


Figure 4 This diagram illustrates the traditional structure of a multi-page application. The browser performs HTTP requests to the web server to obtain an HTML page (HTML pages contain the description of what the page should look like along with the data to display when rendered by a browser). The generation of the HTML code is done on the web server in the presentation layer. As can be seen the presentation layer can interact with other layers around it to obtain the appropriate information to incorporate into its HTML response to the user. In this case the web server returns the current missions of a user. Each new action the user wants to perform will result into another call to the web server with a new HTML response [10].

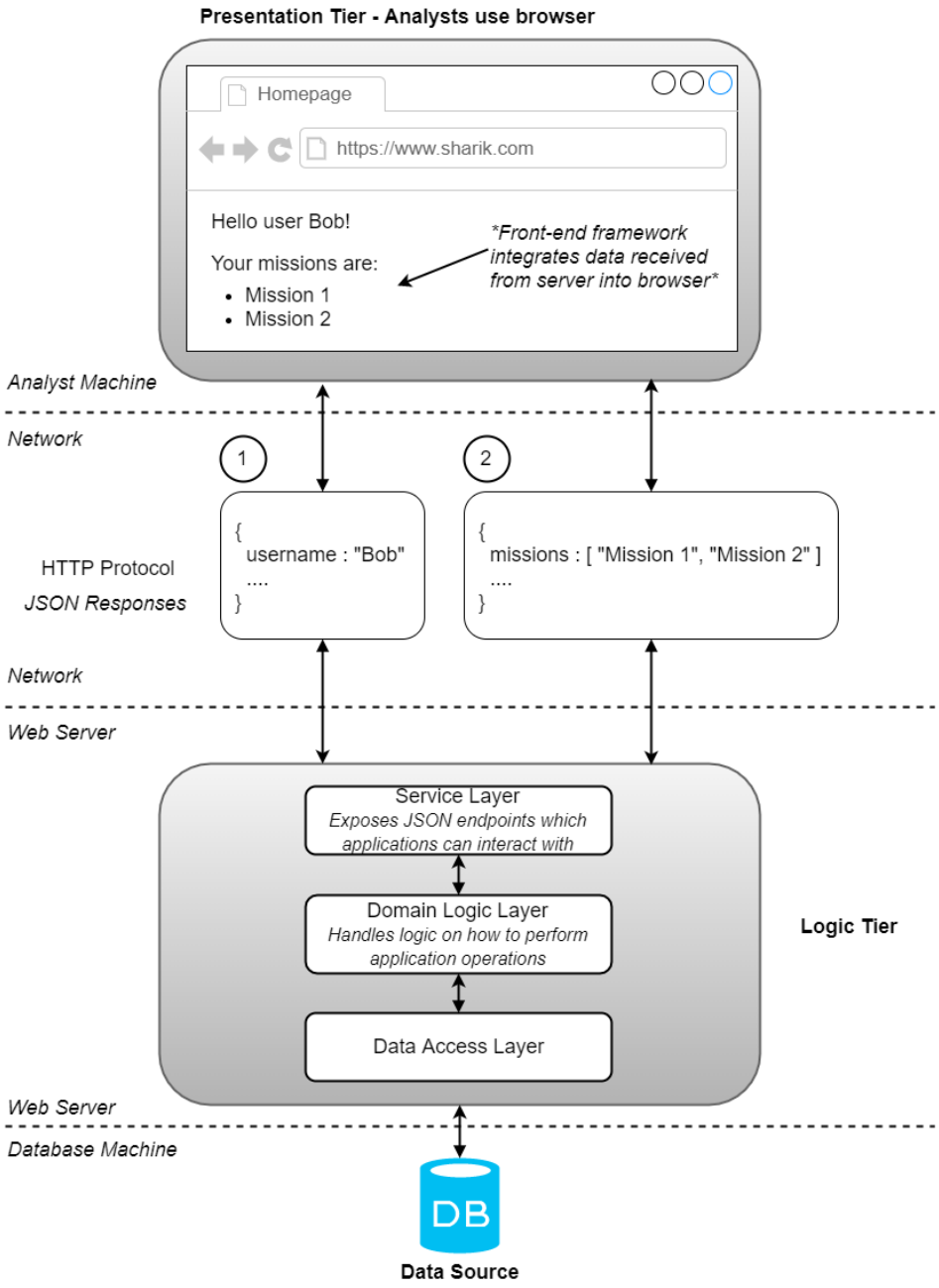


Figure 5 The above figure depicts the single page application approach where the web server returns JSON data responses instead of HTML pages. This approach helps decouple the development cycles of the frontend and backend teams [10]. In the above call sequence, it is assumed the user has already loaded the application with an initial call to server, then to render the homepage, the front-end framework initiates two calls to fetch the data it needs (get the current user's name and the missions they are a part of). Note no other data is requested unlike in the traditional approach which sends back the same HTML structure each time even if only the data changes (e.g. user is now Alice, still send back entire HTML page again like in Figure 4). All HTML manipulation and generation is left to the client-side code, which makes the back-end architecture and tools easier to change since there are no dependencies between the two except for the JSON data formats between them [10].

2.1.1 Selection of Architecture

The engineering team will use the single page application architecture as it provides the most value in contrast to the multi-page approach in regards to the following perspectives [11] [12] [13] [10]:

- **System Extensibility** - The web server will be exposing a set of JSON endpoints. These endpoints expose Sharik's data model but also expose actions that can be performed (e.g. uploading a file). This allows third party systems within DRDC to programmatically leverage these endpoints for future integration with their intelligence communities and software systems which allows for automated software processes (e.g. automatically uploading intelligence reports from department X every night into Sharik without the use of a user interface, only need to know the endpoint where to send data to). This helps maximize the maintainability objective.
- **Familiarity** - The engineering team has more experience developing enterprise level JSON endpoints than multi-page web applications. This factors into how quickly the team can get started and the amount of learning it would take to start producing code and deliverables. This minimizes the initial risk the engineering team takes on, due to the reduced learning curve of the approach, and allow to achieve more project requirements.
- **Productivity & Design Flexibility** - Both back-end and front-end teams are free to work autonomously, decreasing synchronization chokepoints in the development process and allowing maximum flexibility to technology changes (due to the tight timeframe the architecture needs to allow for quick development and failure cycles). The only point where both teams need to collaborate are the JSON endpoints that the back-end developers need to make for the front-end developers to use to enable users of the application to achieve what they need. This helps maximize the maintainability objective.
- **Performance** – The single page architecture allows for more efficient data transfer between back-end and front-end. Furthermore, the architecture allows for a more responsive user interface due to the use of a front-end framework and the single page paradigm. This helps maximize the performance efficiency objective.

The two alternatives presented above serve as a guiding reduction of scope for the latter sections which look at alternatives for each corresponding component within the architectures. This includes the user interface on the client, the front-end frameworks used to create the user interface, the back-end frameworks used to create the web server with, and finally the data sources to be used to store Sharik's data.

The team will include in the project plan on certain allocations of time which will provide certain gateways within the project on whether the single page application architecture is working. If the initial iterations of the architecture do not work, then the engineering team will switch to other technologies or use the multi-page application approach.

2.2 Data Storage Alternatives

This section will explore the means for storing information along with detailing what Sharik will be storing. The data that would need to be stored by Sharik needs to be able to be organized logically and have interfaces to query for information within the data source. These methods need to be supported by existing libraries and use commonly used vendors of storage systems so that the engineering team can leverage these systems as ready to go solutions to decrease development time.

A database meets this need and is also one of the most common patterns for an organization to store their data with [14]. Organizations such as IBM, Microsoft and many other companies use databases [14]. File systems are also the most widely used form of storing files as can be seen through the various implementations of operating systems such as Windows and Linux [14].

Both these databases and file systems have many distinctive designs and implementations. Some common databases will be explored and listed for later selection.

Specific file systems will not be explored as they are not a major component in the design since they need to just store files (no advanced search or other functionality is needed), therefore any standard file system will do for the initial iterations unless issues are discovered [14].

File systems usually do not play a significant role, unless designing large scalable systems for millions of users and is irrelevant especially in the preliminary stages of most software development. This is especially true for applications which are used by a few dozen users through human initiated interactions such as Sharik. In previous industry experience, when designing a real-time streaming system which synchronized filesystems in real-time across machines, the major bottlenecks were in the algorithms and data structures rather than the specific file system at use which have been refined over decades.

2.2.1 Sharik's Data Model

Sharik's data model is relational and contains many relations such as Users, Propositions, Notes, and many others. Each of these relations represents a logical entity within the Sharik environment which will be stored. These relations are also known as tables (see Appendix I for a description of the relational model).

The relational model best suits this type of data because the underlying data models are bounded, known well in advance and contain restrictions and associations with one another which need to be manipulated with precision [15] [14] [16]. Also, the design team is the most familiar with the relational model and has successfully used relational databases in the past. A partial model of Sharik is shown below in a relational database diagram. The model was generated through multiple client consultations and translation of the original design of Sharik's features to the new set of requirements presented in this project.

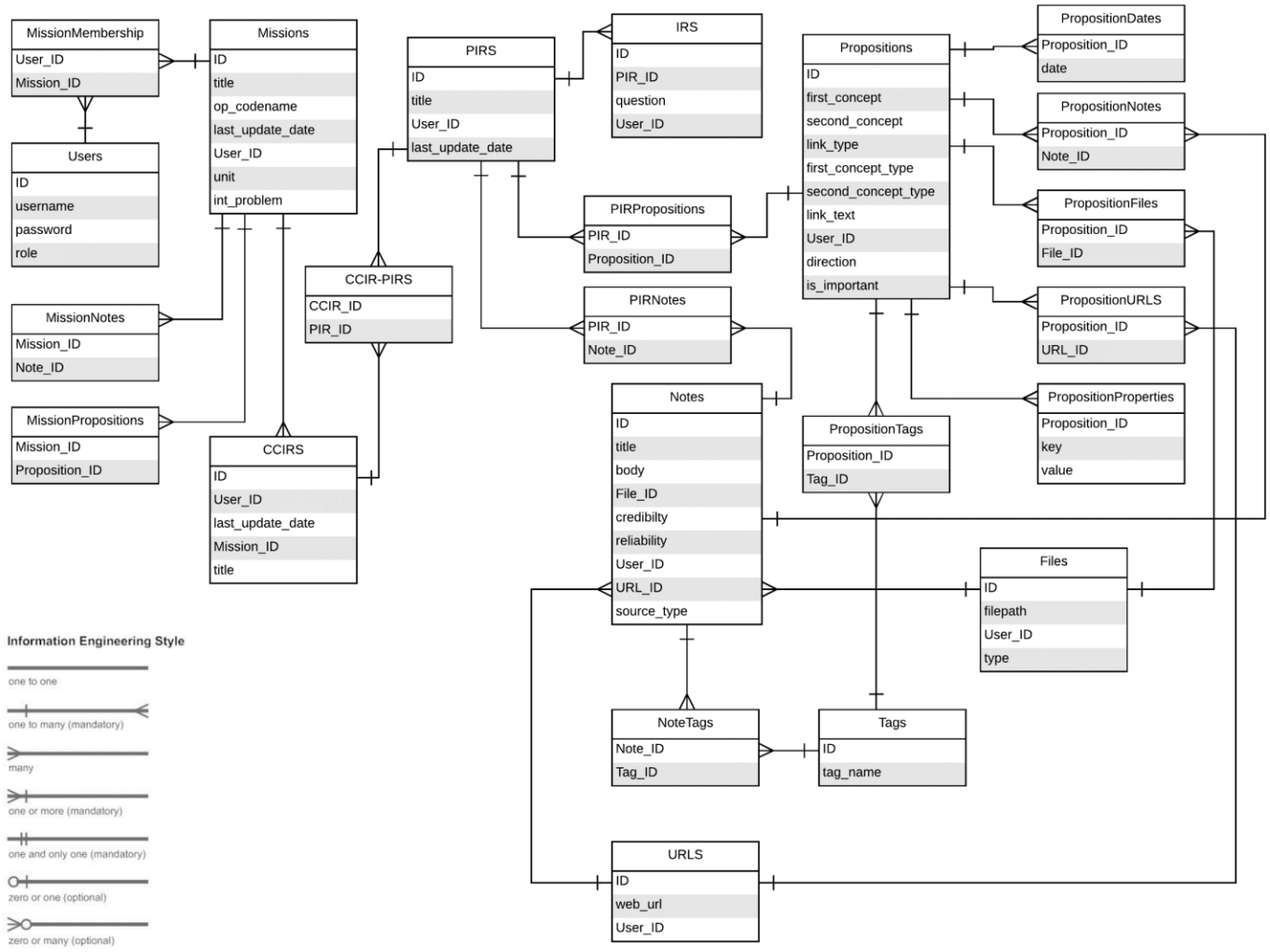


Figure 6 The partial data model for the required features for the new version of Sharik. Note this diagram omits features of Sharik that will not be implemented as they are deemed irrelevant. This diagram maps out how certain tables will be associated with one another. Note the model omits some association arrows between tables for conciseness. An attribute with a capitalization on each word indicates it is a foreign key to another table (e.g. User_ID refers to the ID column of Users). See Appendix J for a description of each attribute and all associations not seen here.

The following will describe each group of tables in the figure above and its purpose within the Sharik environment. The environment can be divided between two models, the model as dictated by DRDC's Intelligence Collection Plan (something we cannot change) and the model used to enable Sharik's features for processing the Intelligence Collection Plan. Note that the following is a high-level summary (for full detail see Appendix J):

- Intelligence Collection Plan Model
    - The Missions table, Critical Commander's Information Requirements (CCIRS) table, Priority Information Requirements (PIRS) table, and Information Requirements (IRS) table are used to represent the hierarchical structure of the intelligence lifecycle as depicted earlier in Figure 1. Each has their own specific fields, consisting mainly of metadata like creation time, who created the element and various other display data like the title of the element. There are also secondary tables which map out ICP element associations (e.g. which PIRs belong to which CCIRs by using the CCIR-PIRS table).
- Sharik Model
    - The Users table stores usernames and passwords for analysts and admins of Sharik and is used when creating new accounts and logging in.
    - The Mission Membership table tracks which user has been assigned to which set of missions. Once assigned to a mission the user automatically is assumed to be part of all sub-elements of the mission (CCIR, PIR, IR and Sharik components).
    - The Notes table represents the concept of a Sharik note which is a text based entry with various metrics and sources attached to it like credibility ratings and file or web based sources through URLs (e.g. website links). There is also an ability to add tags to notes for easier querying later (e.g. look for all notes relating to murder by filtering all notes tagged with “murder”). Many other tables support this table by storing files, URLs and so on.
    - The Propositions table enables analysts to log relationships between intelligence entities (Assassin *killed* Target). The table allows to mark each record with a direction of association (unidirectional/bidirectional) along with associating each proposition with many notes, URL sources, tags, file sources, dates of when the relationship took place, and the ability to add free form properties through a key/value scheme (e.g. key: location, value: Toronto). Many other tables support this table by storing files, URLs and so on.

The above relational diagram is an alternative to modelling the data. Small style decisions can be made to change whether some tables are merged or not, however it is not deemed significant enough to detail in this document as it would provide virtually the same functionality (ability to retain data with certain attributes). Therefore, alternatives for the relational diagrams will only be discussed which provide major functional differences. The next few alternatives will focus around the rest of Sharik’s data storage needs which are Comments, Notifications, Subscriptions, and Auditing.

2.2.2 Data Model - Comments, Notifications, Subscriptions, and Auditing

Below is an alternative for modelling the rest of Sharik's required features (without the slide generation which is covered in a later section) which adds the ability to track changes made to Notes and Propositions. Comments, Notifications and Subscriptions are also shown below even though they have been moved out of the constraints (they are the next highest priority after the constrains so they will be discussed). This alternative shall be called the *coarse grain auditing option*.

Alternative: Coarse Grain Auditing

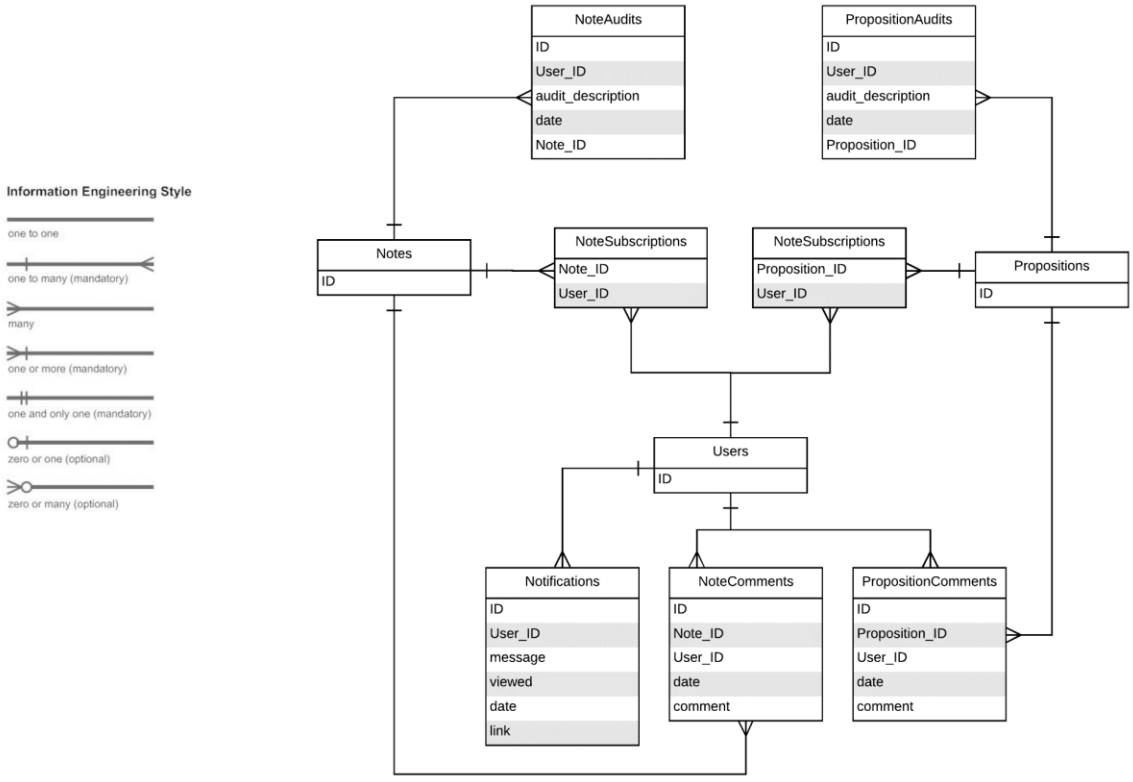


Figure 7 Depicts the Notification, Comment, and Audit models. These tables enable the tracking of changes made to a Note or Proposition. Attributes and other associated tables are removed from Notes, Propositions, and Users to make the diagram clearer.

From the above figure, a user can comment on each Note and Proposition many times through the respective comment tables. Furthermore, the model enables a user to subscribe (subscription tables) to specific notes or propositions and receive notifications for them (e.g. a change was made to a Note). The Notification table keeps track of what notifications have been viewed, when they were issued, and to whom they are addressed to. Also, it provides a link attribute to be able to store links to certain pages or components that the user should go to if clicked.

The audit tables in this alternative only allow for coarse grain auditing – when a change happens on a Note or Proposition, only a description can be stored explaining what happened, which prevents a user interface showing the differences between certain attributes between changes since this is not saved (e.g. can only track "User1 changed Note1", but does not offer exactly which attribute was changed or to what it changed to). This is how Sharik originally operated. The audit tables also track when the change happened through the date column.

An alternative however exists to the Audit model, which is presented below and can track all changes between each attribute. This alternative shall be called the *fine grain auditing option*.

Alternative: Fine Grain Auditing

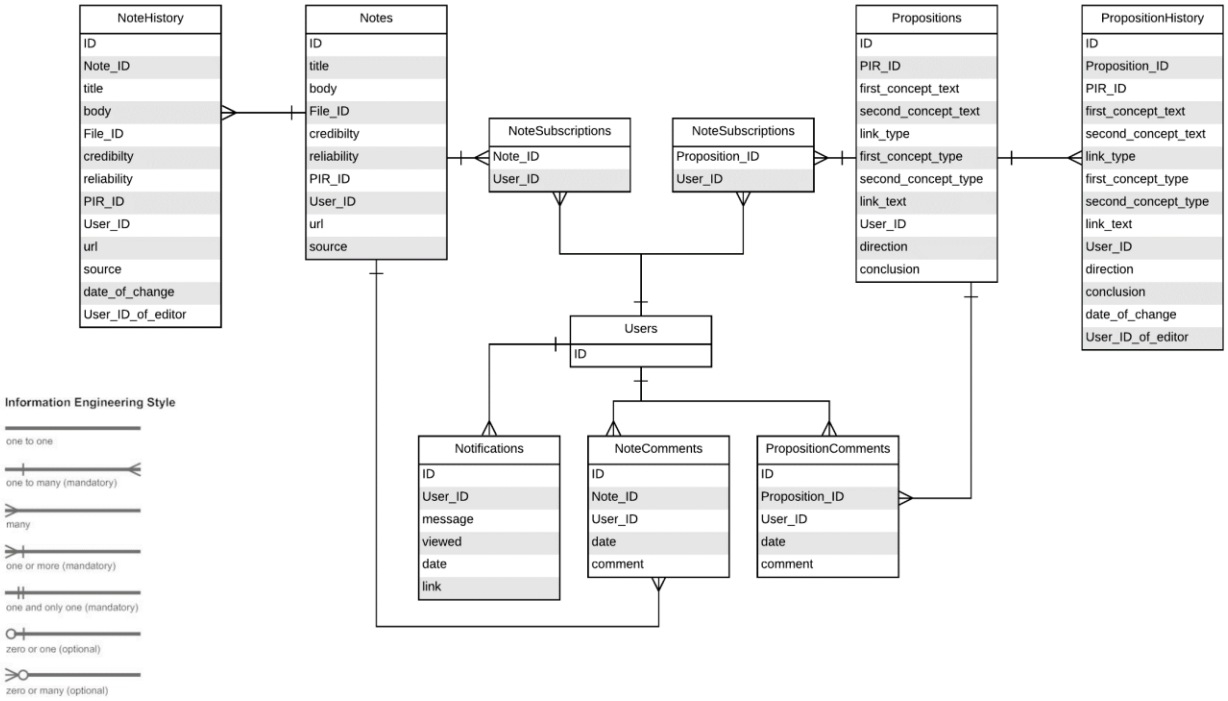


Figure 8 Another alternative for tracking audits for the Notes and Propositions tables. In this alternative two other tables exist which track the full history of a Note or Proposition. The user id of the editor and date of the change are tracked along with all the other attributes of a Note or Proposition. Some tables are removed for brevity as in previous figures. Note this does not enable tracking of all attributes that changed since the above diagram does not show history tables of the other associated tables to Notes and Propositions (e.g. Proposition, Notes) which is explained below. Note the other tables remain the same as in the previous alternative.

This alternative provides the Sharik tool the ability to track all changes made to an attribute in a Note or Proposition. This means the current state of a Note or Proposition is kept in the Notes table and Propositions table, with all its changes up to that point being a separate record in the respective history tables (fine grain auditing). This approach can potentially use a lot of storage space if a Note or Proposition is modified many times as an exact copy of the record is made per modification.

Another concern is that the above alternative does not show history tables for other relations which were associated with Notes or Propositions, and would require investigation on how to make sure if these associated records get deleted how one would then keep track of this within the history tables.

For instance, if we had a Proposition with a file linked to it through the PropositionFiles table, when that file is deleted how would one log the change made in the PropositionHistory table (the file is now non-existent)? This implies that one would have to make sure nothing is ever deleted off the system, adding more complexity for bookkeeping and using more storage space (to be able to recover the exact state and look at a discrete time).

A slight variant on the above approach is to only track the difference between changes to have a smaller storage impact. This is like how Git works (a version control system for software or any other set of files), which uses a lot of complicated data structures and formats to solve the inefficient storage problem [17]. This will most likely require using some database specific functions or software libraries to achieve better storage performance.

2.2.3 Database Alternatives

To be able to create and store these relational models a relational database will be used since they are designed specifically for this task [14] [16]. Due to the constraints, only relational database vendors which are free will be considered for storing Sharik's data.

There are few popular options that exist for relational database alternatives as most companies stick to a select few as seen from experience in large companies (most Fortune 500 companies use DB2 or Oracle) with other companies going with the free MySQL database. The three alternatives that will be considered which are free and are the top three relational databases as ranked by a set of well-defined metrics are (as of November 2017) [18] [19]:

1. MySQL (Score: 1322)
2. PostgreSQL (Score: 380)
3. SQLite (Score: 113)

The rankings above used the following metrics to rank and score them [20]:

- number of mentions of the system on websites (Google, Yandex, Bing)
- Google Trends statistics
- frequency of technical discussions about the system (Stack Overflow, DBA Stack Exchange)
- number of job offers, in which the system is mentioned (Indeed, Simply Hired)
- number of profiles in professional networks, in which the system is mentioned (LinkedIn, Upwork)
- relevance in social networks (number of Tweets)

Note that the overall rank (compared to databases of all types) of MySQL is 2nd, PostgreSQL is 4th and SQLite is 7th. The scores indicate the relative popularity of the system, so for example MySQL is around 3.5 times more popular on each individual metric than PostgreSQL [20].

2.2.4 Data Storage Alternative Selection

From the above sections, the engineering team will implement the core Sharik model presented in Figure 6 as it provides all the required relations to meet the core features of the project such as the ICP elements, users, notes and propositions. The engineering team will further ideate with the client on the specific columns that the data model needs to store and what type of data they can take on as the project progresses.

The next choice of alternative for the team is to do the coarse grain auditing approach. Due to the level of difficulty and unknowns presented in the fine grain auditing option the team opts to select the coarse grain auditing option. This is to reduce risk within the project and still be able to abide by the auditing constraint.

The fine grain option would bloat the database with many history tables and possibly add more code needed to maintain those tables and bookkeeping processes, adding more points of failure within the system. However fine grain auditing would be beneficial to add in the future to Sharik as it would provide the ideal auditing granularity for an intelligence application as mentioned in client consultations (accountability and traceability of user actions around sensitive intelligence). This however was not deemed important in the requirements for the focus group trial, but noted as a required feature for future iterations of the tool (outside of the current scope).

MySQL will be used to store Sharik's relational model due to it being ranked the highest amongst the free database vendors and having been used by DRDC successfully in the past. The ranking as mentioned in 2.2.3 was based off metrics which indicate how well known the database is and supported by the software development community. Choosing a database that is widely used helps the engineering team leverage already existing documentation, best practices, as well as providing the client the maximum opportunity to find developers for the future to continue maintaining the system the engineering team will deliver (maximizes maintainability objective).

2.3 Front-end Frameworks

In web development, the front-end refers to the parts of the code that manifest in visible web elements for the user. Creating rich and dynamic user interfaces is possible with just plain

JavaScript, but adopting a framework dramatically decreases development time by providing constructs for common programming idioms and hiding implementation details of prolific web design patterns.

The enormous amount of innovation occurring in the current ecosystem surrounding JavaScript has resulted in a wealth of choice in terms of workflow and design philosophies but has also made it difficult to find consensus on best practices. Due to the amount of fragmentation, we only explore a subset of the most popular frameworks.

Table 1 This table provides a quick overview of the frameworks we chose to explore in detail.

Framework	Release Year	GitHub Repo Stars as of 17/11/08
Angular 2	2016	29,793
React	2013	80,560
Vue	2014	73,079

2.3.1 Feature Scope

Front-end frameworks normally are specialized to provide libraries for the view component of an application, but some frameworks are designed to handle web functionality across a wider range of domains. Choosing a more fully featured framework may allow us to conserve mental effort by freeing us from the obligation to choose other components of the architecture, but may reduce the amount of flexibility in case we require a more custom design for our needs. In this case, we would have to find other components to perform things like form validation in the case of React and Vue.

Table 2 [21] The following table shows the various features of the frameworks we chose to explore.

	Angular 2	React	Vue
View/Templating	✓	✓	✓
Router	✓	✓	✓
Form processing	✓		
Form validation	✓		
HTTP communication	✓		

2.3.2 Performance Benchmarks

We compare the relative performance of each framework to a vanilla JavaScript (plain) implementation which is depicted in the far-right column as the best achievable performance. Vue.js is the most performant of the frameworks while Angular 2 is the least.

Figure 9 [22] The following table shows performance results for the frameworks we chose to explore with a lower level implementation of the performance tests written in plain JavaScript as reference. Each performance test involved an update of an UI element or a data element in response to a keyboard key press.

Performance Test \ JavaScript Framework	angular v4.1.2-keyed	react v15.5.4-keyed	vue v2.3.3-keyed	vanillajs-keyed
create rows Duration for creating 1000 rows after the page loaded.	193.09 7.88 (1.39)	188.93 10.93 (1.36)	166.68 8.63 (1.20)	138.47 5.80 (1.00)
replace all rows Duration for updating all 1000 rows of the table (with 5 warmup iterations).	197.37 5.25 (1.33)	201.03 6.40 (1.36)	168.49 4.98 (1.14)	148.01 4.47 (1.00)
partial update Time to update the text of every 10th row (with 5 warmup iterations).	12.98 4.47 (1.00)	16.48 2.30 (1.03)	17.33 2.90 (1.08)	14.15 4.73 (1.00)
select row Duration to highlight a row in response to a click on the row. (with 5 warmup iterations).	3.39 2.30 (1.00)	8.76 3.37 (1.00)	9.31 1.66 (1.00)	10.10 4.68 (1.00)
swap rows Time to swap 2 rows on a 1K table. (with 5 warmup iterations).	13.43 1.04 (1.00)	14.66 0.90 (1.00)	18.29 1.52 (1.14)	11.43 1.13 (1.00)
remove row Duration to remove a row. (with 5 warmup iterations).	46.13 3.18 (1.09)	47.22 3.20 (1.12)	52.63 2.69 (1.24)	42.82 1.91 (1.01)
create many rows Duration to create 10,000 rows	1946.02 41.79 (1.46)	1852.36 29.03 (1.39)	1587.52 33.89 (1.19)	1331.13 22.16 (1.00)
append rows to large table Duration for adding 1000 rows on a table of 10,000 rows.	324.63 10.08 (1.13)	345.62 10.40 (1.20)	399.46 10.98 (1.39)	295.31 12.82 (1.03)
clear rows Duration to clear the table filled with 10.000 rows.	379.94 11.25 (2.17)	398.38 8.25 (2.28)	254.51 5.03 (1.46)	174.77 4.19 (1.00)
startup time Time for loading, parsing and starting up	84.34 2.64 (2.08)	69.98 2.85 (1.73)	56.55 2.48 (1.39)	40.55 9.54 (1.00)
slowdown geometric mean	1.31	1.30	1.22	1.00

2.3.3 Learning Curve

Determining learning curve is difficult as it depends heavily on gathering many opinions from many different developers. Thankfully we have the 2016 and 2017 State of JavaScript survey results [23] [24], which was answered by over 9000 and 20,000 JavaScript developers respectfully. The survey results show that interest in Vue.js is growing the fastest, but React.js

currently holds the greatest amount of interest. Angular 2 lags in terms of interest in all regards and causes the most amount of developer dissatisfaction. We currently have one member with some Vue.js experience.

In each of the following plots, the results of the 2016 and 2017 surveys are shown from left to right order percentages of people who 1) Never heard of it, 2) Heard of it, but not interested, 3) Heard of it, would like to learn, 4) Has used it, would not use it again, and 5) Would use it again.

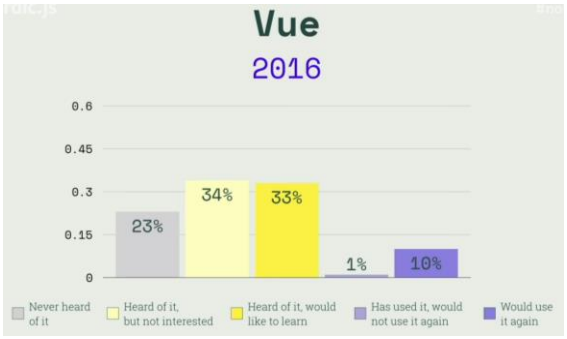


Figure 10 Usage of Vue.js in 2016

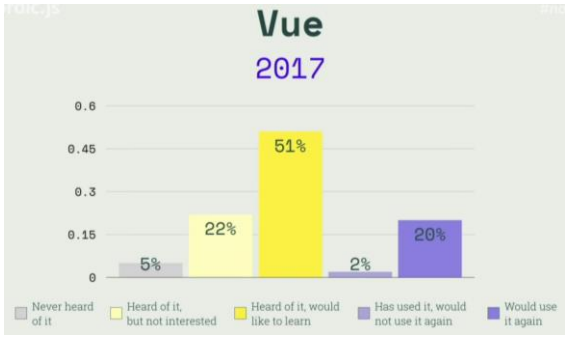


Figure 11 Usage of Vue.js in 2017

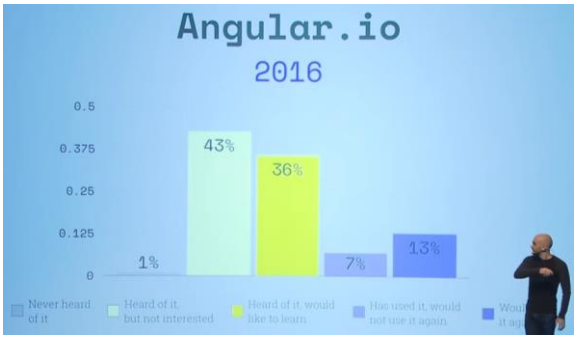


Figure 12 Usage of Angular 2 in 2016

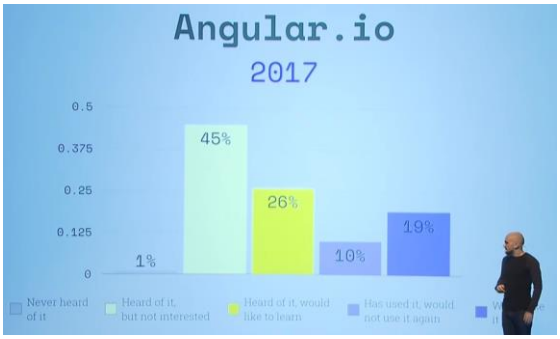


Figure 13 Usage of Angular 2 in 2016

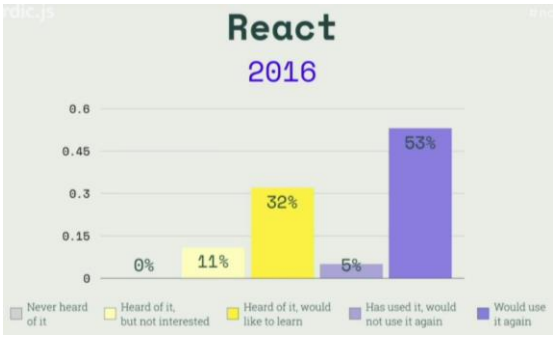


Figure 14 Usage of React.js in 2016

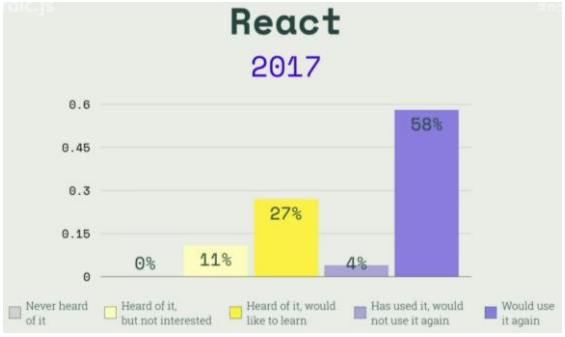


Figure 15 Usage of React.js in 2017

We also cautiously refer to the *Comparison with Other Frameworks* located on the Vue.js website [25], which lists some of the difficulties associated with React.js such as learning JSX to replace HTML and learning build systems to replace simply inserting a script tag to reference Vue.js. We recognize the possibility of bias, but are assured by the involvement of the React.js (the competitor) community in writing this document on GitHub.

2.3.4 Selection

We opt to move forward with using Vue.js because of its above-average performance benchmarks to help meet our time behavior requirements, and significant community interest in the framework and prior experience within our team to help meet learnability requirements. Even though React is currently more popular, when totalling the number of people who "Would use it again" and the people who "Heard of it, would like to learn", Vue's popularity is on par with React having a total of 85% and Vue having a total of 71%.

2.4 CSS Frameworks

CSS frameworks are related to front-end frameworks in that they deal with web elements that are visible to the user, but are different in that front-end frameworks deal with the mechanism of operation for UI elements whereas CSS frameworks deal with the appearance of UI elements.

Using a CSS framework in favor of plain CSS gives developers access to pre-styled UI elements to reduce the amount of time needed to achieve a certain aesthetic. As with front-end JavaScript frameworks, there is a wealth of choice, but in contrast, one CSS framework is pre-dominant – Twitter's Bootstrap.

As CSS frameworks deal with mainly aesthetics, the use of one over the other is highly subjective. There does not appear to be a best practices guide so the approach this design team took was to gather the results of a Google query of "top CSS framework" and aggregate the votes. The following table was constructed after surveying the following 8 blogs:

1. https://medium.com/@thomasmarciniak/top-css-frameworks-to-follow-in-2017-51d283dd00fb
2. https://three29.com/best-css-frameworks-2017/
3. http://www.discoversdk.com/blog/top-10-css-frameworks
4. https://www.markupbox.com/blog/top-5-popular-css-frameworks-of-2017/
5. https://www.catswhocode.com/blog/lightweight-css-frameworks-2017
6. http://www.cssnewbie.com/12-awesome-css-frameworks-for-your-next-project/
7. https://hackernoon.com/top-5-most-popular-css-frameworks-that-you-should-pay-attention-to-in-2017-344a8b67fba1
8. https://www.keycdn.com/blog/front-end-frameworks/

Table 3 This table contains the number of mentions of each CSS framework across the eight blogs that were surveyed. Bootstrap is both the oldest and most popular framework of the top five frameworks mentioned.

CSS Framework	Release Year	GitHub Stars as of 17/11/08	Mentions in First Page of Blog Post Results When Googling "top CSS framework"
Pure	2013	17693	7
Bootstrap	2011	117415	6
Bulma	2016	21395	5
Materialize	2014	29720	5
Milligram	2015	6553	5

2.4.1 Selection

We opt to move forward with Twitter Bootstrap due to its much higher popularity and community support as well as our team's prior experience with Twitter Bootstrap that push it ahead of all other frameworks in terms of learnability and maturity.

2.5 Back-End/REST API Frameworks

Server-side processing attempts to reduce the load on the client or browser by performing expensive computation before sending the results to be rendered. This can be in the form of static HTML pages being sent or simply the results of a database interaction. Like the previous frameworks discussed, back-end frameworks work to reduce the amount of development time through abstracting common design patterns used by developers.

With the much greater diversity of frameworks used on the server-side compared to the client-side due to multiple viable languages for server-side rendering, we limited our search to one open-source framework per language. The chosen frameworks were heavily biased toward frameworks we had prior experience followed by their popularity among the community. Less popular frameworks that were successors of a previous framework were chosen over their predecessors.

Table 4 This table contains the most popular back-end framework for each of the most popular server-side technologies currently used in the web. Popularity ranking was determined from a third-party source that ranked frameworks based on their GitHub stars and the prevalence of questions pertaining to the framework [26]. To get the best possible representation of languages used in back-end frameworks, we consulted a third-party source [27].

Framework	Represented Language	Framework Popularity Ranking	Language for building REST API Popularity	Release Year
Ruby on Rails	Ruby	3	6	2005
Django	Python	6	5	2005
Laravel	PHP	8	2	2011
Spring	Java	9	3	2002
Express	Node.js	10	1	2010

2.5.1 Decision Criteria

From the shortlist of frameworks, we borrow selected criteria from Mozilla on how to best choose a framework. [28] We rank them based on our own objectives with the accompanying rationale:

1. Effort to learn:
   We emphasize this the highest as we have a limited amount of time and relatively no overlap in terms of web development experience. To get every member of our team to the same level of expertise, we require easy frameworks to learn. Existing experience must also be considered because having other teammates to teach the framework will also significantly lower the barrier to entry.
2. Whether or not the framework encourages good development practices:

As we are putting the emphasis on speed, we realize we may be making sub-optimal infrastructure and component decisions. Therefore, we hope to develop modular code that can be swapped in and out depending on future needs.

3. Batteries included vs. get it yourself:
   Due to the limited time and the lack of experience with professional web development within the team, frameworks that include by default many tools and libraries are preferred to reduce amount of time for new features.

Table 5 [29] [30] This table contains a qualitative evaluation of each framework regarding the three criteria outlined above compiled from various web sources.

Framework	Effort to learn	Encourages Good Development Practices	Batteries Included
Ruby on Rails	High (We have SMEs on our team)	Yes	Yes
Django	Low	Yes	Yes
Laravel	Low (We have SMEs on our team)	Yes	Yes
Express	Medium	No	No
Spring	Low (We have SMEs on our team)	Yes	Yes

2.5.2 Selection

We opt to move forward with Spring due to the greater proficiency with Java present on our team than any other language, which is an immense factor in its learnability, and the long history of the framework and language, which contributes to its maturity. We choose not to consider Ruby on Rails due to a previous member's poor experience with the framework and not to consider Django or Express due to the complete lack of experience in our team. Most of the mentioned frameworks have similar levels of popularity and follow similar design principles, so these factors are not considered in this decision.

2.6 Web Server

A web server is the component of a website that handles HTTP requests, allowing a web application to send and receive data.

2.6.1 Selection

Of the 1,815,237,491 websites surveyed by the Netcraft monthly web server survey, the top open-source web servers in terms of market share were Apache at 18.23% and NGINX at 17.48% [29]. However, if we are to move forward with Java-based server-side technologies, we must use Java application servers such as Apache Tomcat or JBoss. We opt to move forward with either Apache Tomcat if we are to pursue Java-based server-side languages or NGINX if we are to pursue other server-side languages due to team members possessing prior experience with those two specific web servers aiding its learnability.

2.7 Operating System

The operating system is the software that all other components mentioned so far runs on. We posit that there are two alternatives for the choice of operating system: Windows or Linux. We exclude others because either they are too niche to be considered or are proprietary and would incur additional costs without any meaningful benefit.

2.7.1 Selection

From experience, Linux has proven to be a more developer friendly environment to host web applications. Many configuration management tools and virtualization technologies that expedite application deployment and setup are catered to Linux and the design team is more familiar with Linux than Windows contributing to its learnability. Linux is also easily accessible on most cloud offerings such as AWS and Digital Ocean while Windows is not making it better for operability. This will be useful for providing a live ongoing demo to interested parties, for the design team to easily setup their developer environments, and for keeping configuration differences between development environment and the live production environment for Sharik minimal. We opt to use Ubuntu as it is the most popular distribution of Linux and thus will have the most community support [30].

2.8 User Interface

In this section, user interface design alternatives were brainstormed, refined, and evaluated through a heuristic evaluation method that will be described in detail. Experienced UI designers often conduct a heuristic evaluation on preliminary designs, revise the design prototypes, then conduct usability testing [31]. Furthermore, literature suggests that heuristic evaluation followed by revisions improves the design by 50% every iteration, which the team believes to be adequate at this stage [31]. Due to time considerations, only heuristic evaluation is included in this document. Usability testing will be performed before the Design Critique, as detailed in the Project Plan section at the end of this document.

The user interface of the proposed solution was divided into high-urgency and low-urgency elements to be designed. This decision was based on how many iterations were likely to be needed when designing the given UI element, as well as client priorities. Further, some aspects of the user interface have few variants in practice (e.g. login screens), among which none add significant value to this project. The following subsections address the most urgent UI design decisions, such as the method for data entry and the method for slideshow creation. While the design decision on placement of UI elements within the webpage will be among the most apparent to the end user, deciding between the few permutations in which this can be done is not an urgent priority compared to the design work of the UI elements themselves. Therefore, for the purposes of designing the UI elements in context, we developed one possible layout of the webpage, depicted in Appendix E.

The CMap is a high-priority aspect of the user-interface. Despite this, the prototyping of the CMap has been deferred because the Sharik user study did not reveal significant problems with its realization in the initial version of Sharik. Emphasis has instead been focused on more problematic features of the initial Sharik version, or features that were not present in the initial version.

Table 6 Urgency classification of the design of user interface elements. Superscripts indicate the reason for deferral (1: Choosing among divergent solutions offers minimal value to the solution, 2: Low priority based on client feedback, 3: Optional feature, 4: Requires further discussion with client)

High Urgency	**Defer to Design Critique**
Presentation creation	Editing of entities[1]
Data entry	Note display tab[1]
Post-creation editing of notes and propositions	Administrative user management[1]
Representation of mission hierarchy	Login screen[1]
Filesystem UI	CMap[1]
	CMap timeline[2]
	Help button[2]
	Space for pictures, PDF previews, etc. in propositions[1,4]
	User profile[1,4]
	Commenting system (i.e. subscription and viewing comments)[1,2]
	Inputting of types for properties[1]
	CMap filtering of propositions and entities[1]
	Header bar (i.e. My PIRs, notifications)[1,4]
	Revision history[3]

To prototype the UI elements of the proposed solution, we created low-fidelity prototypes, which are comparatively fast easy pen and paper designs consisting of drawings, or low-fidelity digital designs (called *wireframes*) depending on which option was quicker. Early use of low fidelity methods of prototyping user interfaces makes modifications less time-consuming during usability testing. Furthermore, users who are performing the evaluation give more substantive feedback regarding the functionality of the prototypes rather than on minute details such as typeface [31].

2.8.1 Data Entry Ideation

The existing user study on Sharik conducted by DRDC [32] identified that the data-entry step in the original prototype of Sharik was a bottleneck in the workflow of analysts. The existing sidebar used for data entry had two text fields that were often confused. Additional metadata such as attachments could not be specified via the sidebar, resulting in analysts spending additional time specifying such metadata using the point-and-click form of data-entry. Furthermore, the existing data-entry methods did not facilitate analysts with providing data in the correct format, or with autocompleting pre-existing entities stored in the system.

To address the challenges with data-entry in Sharik, an ideation session was conducted in which we diverged on possible methods for the data-entry of propositions and notes. After the ideation session, we identified four general categories of options for data-entry elements:

- "Breadcrumb"/Command-line (Figure 16, Figure 17, Figure 18, Figure 19): The analyst types either a note or a proposition within the single provided textbox. As the user types,

the system infers the semantical meaning of the text and provides drop-down lists of autocompletion results. Based on the options selected in the drop-down lists, the system can determine whether the text represents a proposition or a note, and will store it as such. Metadata such as attached files can be specified using an uncommon token (i.e., a keyboard character that represents a concept) such as "@" or "#" followed by the type of metadata and the content of the metadata. Because the metadata type is specified, the system can provide autocompletion for metadata.

- Point-and-click form: The analyst enters data in labelled form fields by selecting the field with the mouse and then typing the content. The client expressed that this option would impede the data entry of analysts too severely. Therefore, we will not consider this option any further.
- Through CMap (Figure 21): Analyst uses the mouse clicks and drags to create and modify propositions in place in the CMap; note entry would be implemented using one of the other 3 options.
- Free-form text (Figure 20): This option is most like the existing quick-access functionality in Sharik [1]. The analyst types either a note or a proposition within the single provided textbox. If the text starts with a predetermined token such as "NOTE" or "Note", the system is to assume that a note follows. Otherwise, the system is to assume that a proposition follows. The system would assume the note title is on the same line as the "NOTE" token, and that the content of the note follows on a new line. Metadata such as attached files would be input using a point-and-click form.

We designed the four method categories to support the input of all note and proposition metadata (e.g., date, place, attached files) along their content. We envision that proposition and note editing will be performed in the same fashion as initial input, rather than via a limited-functionality quick entry for input followed by a full-functionality form-based input method for editing. By enabling the input of metadata, we will be able to eliminate this duality, thereby rendering the user interface more consistent, and less reliant on point-and-click based input which the client wished to minimize [32].

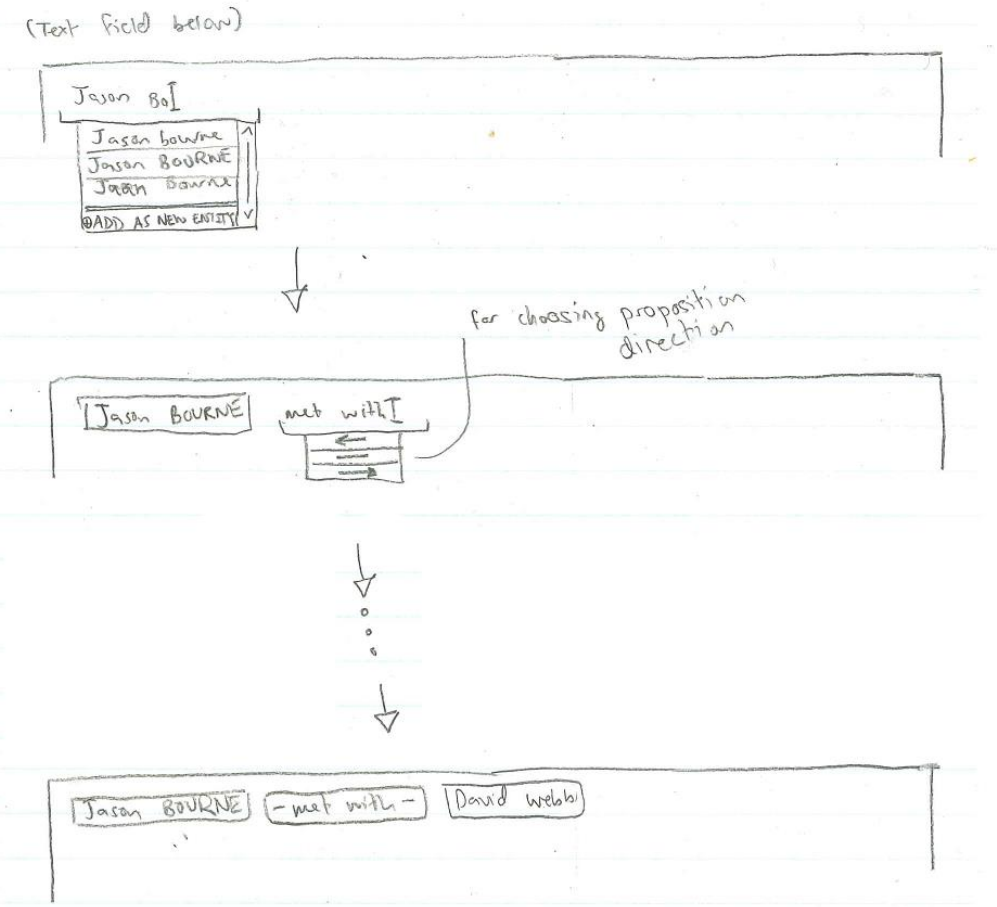


Figure 16 A proposition is entered through command-line style. Here, an analyst begins the creation of a proposition by typing in the name of an entity, which is autocompleted. This process continues with the entry of a linking phrase and a second entity.

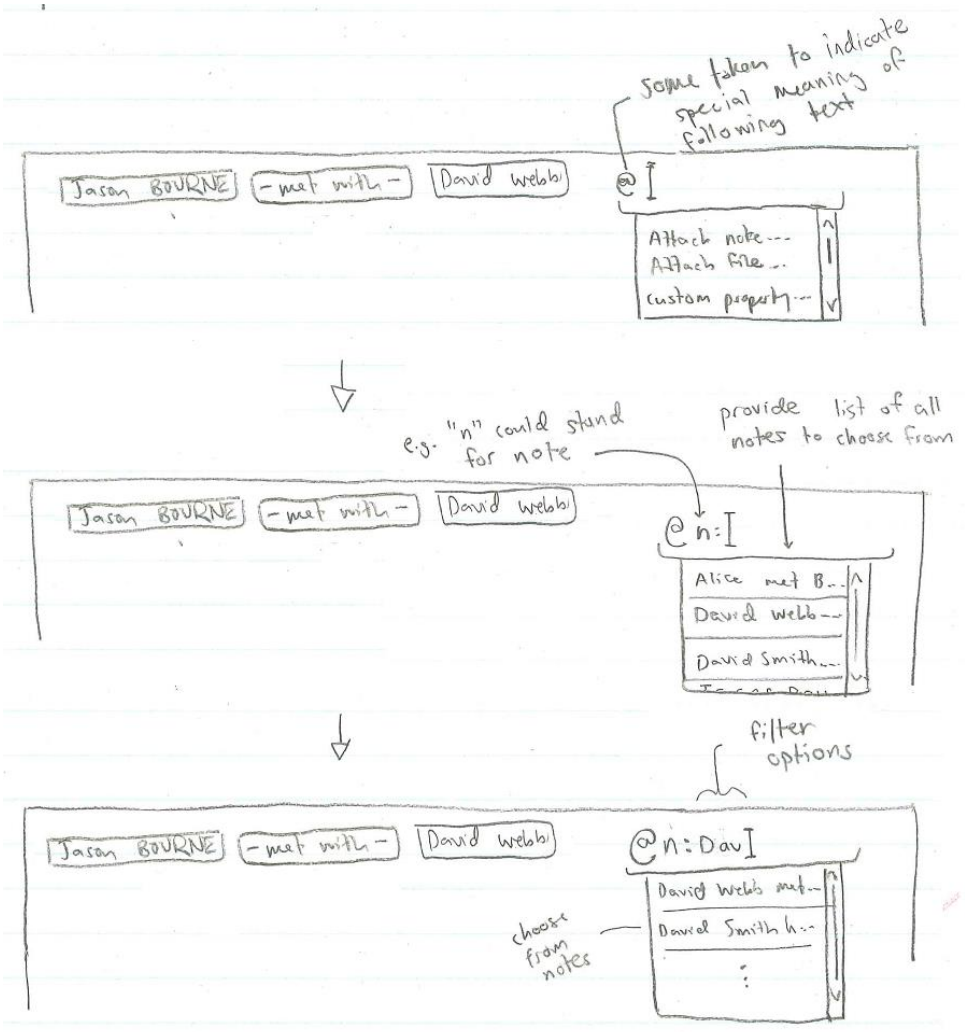


Figure 17 A note is attached to a proposition within the command-line style. The analyst types in a special token to prompt the autocompletion of existing notes.

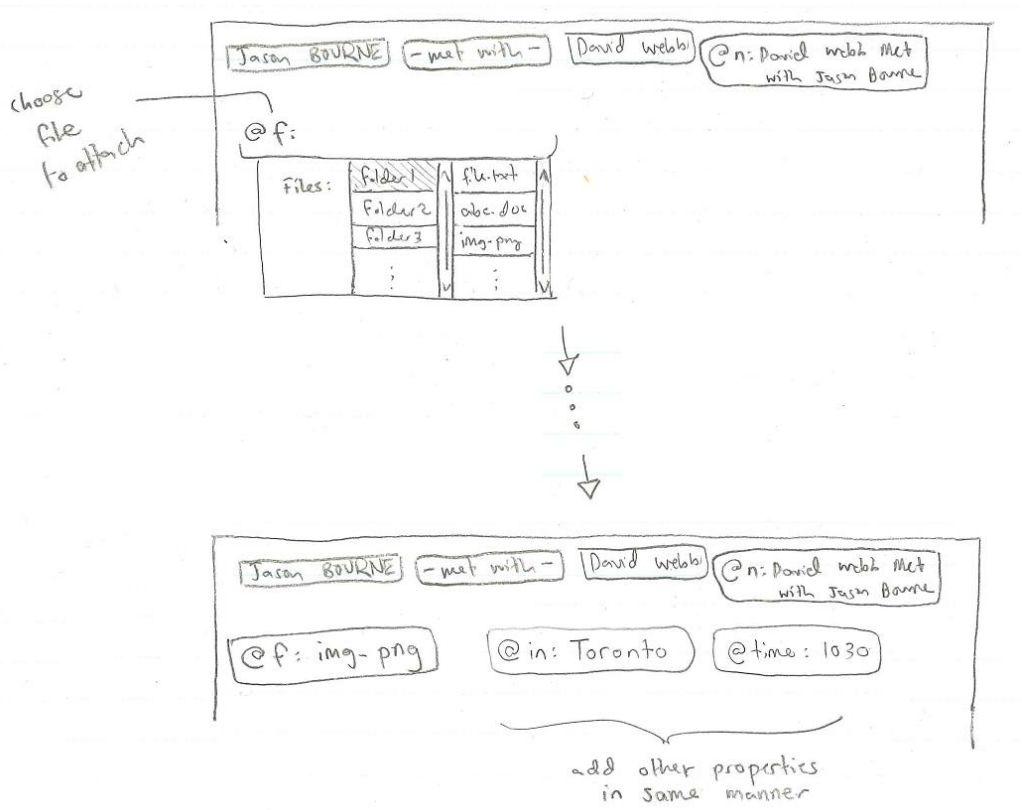


Figure 18 A file and additional properties are attached to a proposition within the command-line style. Autocompletion operates in a similar fashion to the autocompletion for notes.

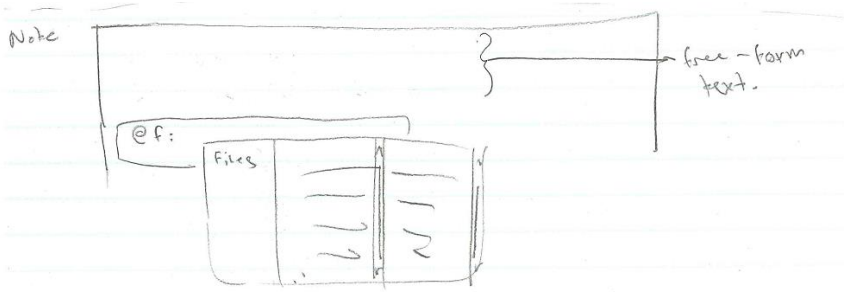


Figure 19 A note is entered through command-line style. The system can infer that the input is a note, because the mandatory concepts and linking phrase of a proposition have not been included.

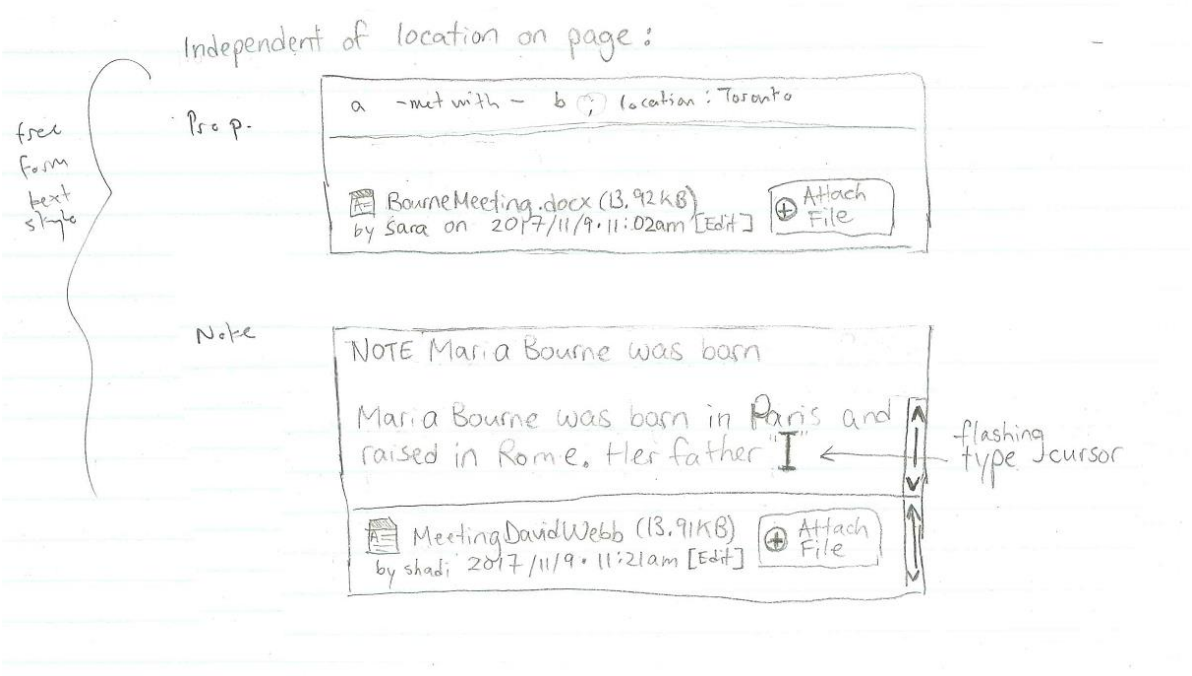


Figure 20 The free-form scheme for proposition and note entry operates using a free-form text box. When an analyst types "NOTE", the system will automatically register the input as a note (otherwise it is registered as a proposition). Text on the same line as "NOTE" is registered as the title of the note. The syntax of the proposition would be an adaptation of the proposition syntax from the initial version Sharik.

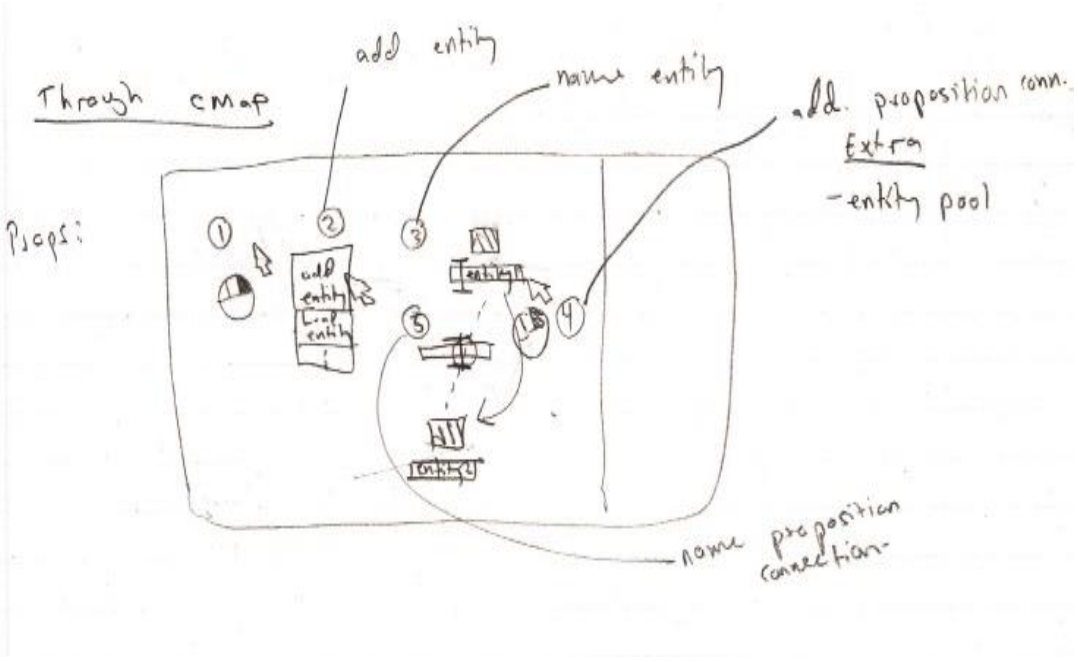


Figure 21 Process Diagram for proposition entry through the CMap. The analyst would right-click inside of the CMap window to bring up the pop-up menu (e.g. where they can choose to add an entity), type in the entity name, right-click the second entity to connect to, then the type in a name for the proposition connection.

2.8.2 Slide Creation Ideation

Collaborative slide creation was one of the two most important priorities for DRDC. Following an ideation session, we identified three different alternatives for slide creation. Table 7 compares between these three different slide creation options. The options themselves are roughly depicted in Figure 22, Figure 23, and Figure 24. While the options refer to PPT files, Microsoft PowerPoint software, and Microsoft OneDrive, these technologies merely serve as an example implementation of those options. Within each option, there may be alternative technologies that achieve the same goal of storing slides, editing slides, and sharing content, respectively.

Should the Custom In-Browser slide creation option (Figure 24) be chosen, there are three sub-alternatives that relate to the way in which propositions and notes are linked within the presentation slides. Once a proposition or note is dropped into the custom slides, it (a) becomes plain text, (b) becomes non-editable from the Slide window, but the user can remove it by deselecting the proposition or note currently shaded in the dropdown window as shown in Figure 24, or (c) is completely linked to the Sharik database, and updates whenever the user refreshes the web page in the browser.

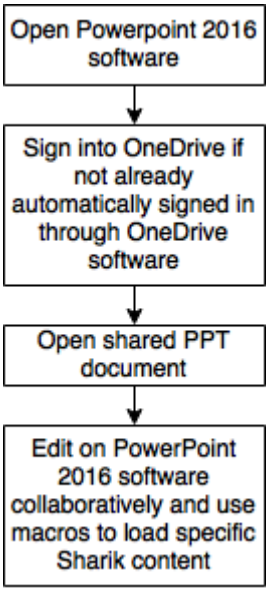


Figure 22 PPT + OneDrive: The analyst opens PowerPoint 2016 software that is pre-installed on his/her computer, as this version allows for collaborative editing. Depending on security settings on their laptop, the analyst may be required to sign into his/her pre-created OneDrive account to connect to the collaborative environment from inside PowerPoint 2016. Once signed in, they create or open a shared PPT from within a shared OneDrive folder (dedicated to Sharik presentations) that all analysts have access to. Once the PPT is open in PowerPoint 2016, an analyst can use macros to load specific Sharik content (e.g. propositions, notes) into the slides. All changes will be seen real-time by collaborating analysts. More generally, this option could be implemented using alternative online slide creation software or libraries which could be hosted internally by DRDC.

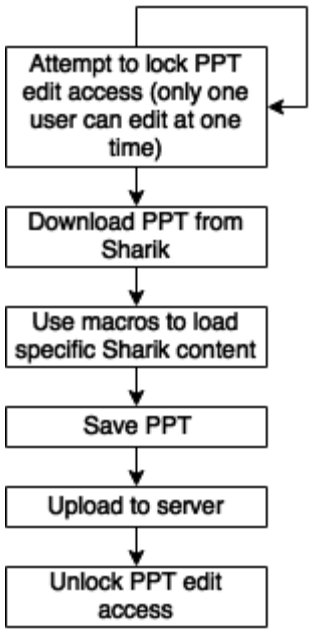


Figure 23 PPT + Lock: Multiple analysts may want to edit one PPT, but only the first analyst who clicks to open the PPT file from Sharik will be able to "lock" (i.e. temporarily gain exclusive access to) the PPT and be able to edit it. Sharik grants this first analyst access to the PowerPoint, downloads the file to the analyst's laptop and automatically opens it in the PowerPoint software. The analyst can use macros to load specific Sharik content (e.g. propositions, notes) into the PowerPoint. Once done editing, the analyst saves inside PowerPoint, closes the file, and uploads it back to the Sharik server. This "unlocks" the PPT, re-enabling edit access for other analysts.

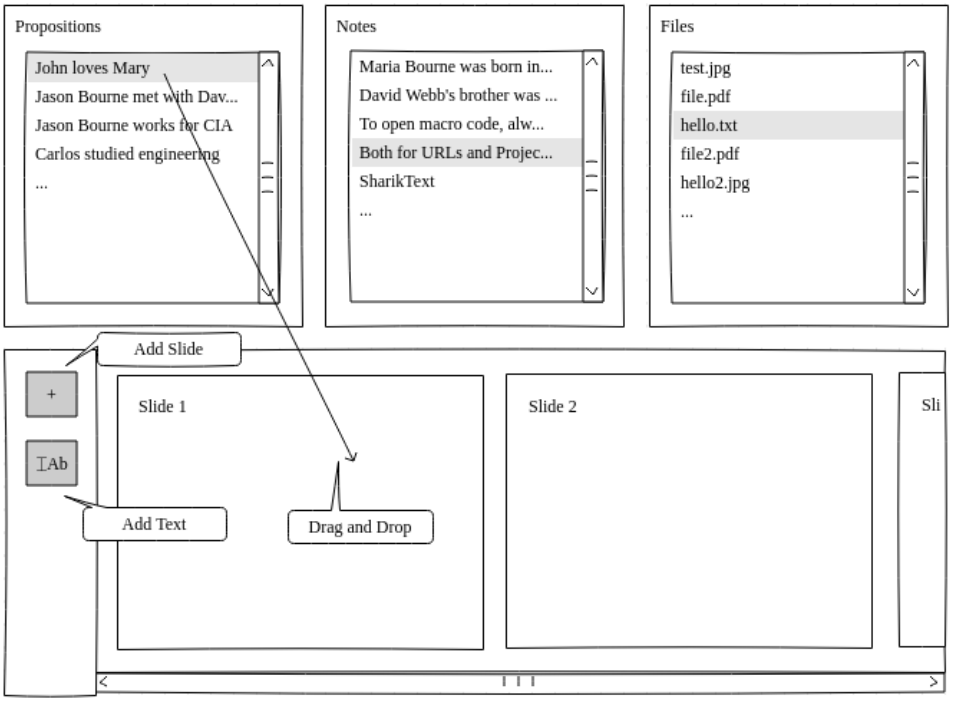


Figure 24 Custom In-Browser + Slide Lock: Propositions, notes, and files are displayed, and can be dragged into slides in the slide deck. Text boxes can be added by clicking the "Add Text" button and then typing in content.

Table 7 Slide creation options: “Real-time” refers to real-time collaboration between multiple analysts on editing slides (as opposed to file or slide-granularity locking). “Does not need (new) desktop software” refers to e.g., the PowerPoint 2016 software that allows for real-time slide collaboration. “Does not need online sign-in (security)” refers to the need to sign-in to an external server such as Microsoft OneDrive to enable real-time collaboration on slide creation and editing. Lastly, “Many design features” refers to the presence of many editing features available for creating and editing slides (e.g. positioning textboxes, language check) within a slide deck. Many features that are normally available in PowerPoint software will not be available in the Custom In-Browser option. (* OneDrive would require an online sign-in, however, open-source online slide-creation tools could enable the real-time collaboration of analysts if they have sufficient support for Sharik integration).

Feature	PPT+OneDrive	PPT+Lock	Custom In-Browser
Real-time	✓	✗	✓
Does not need (new) desktop software	✗	✗	✓
Does not need online sign-in (security)	✗*	✓	✓
Many design features	✓	✓	✗

2.8.3 Slide Creation Data Model

Based on the alternatives for the custom in-browser feature, the plain-text alternative's data model would have columns for the slide number and the content. Alternatives in which notes, and propositions are linked to slides (i.e., sub-alternatives (b) and (c) as discussed in Section 2.8.2) would require similar data to data depicted in Table 8 to store slide information. The client has specified that the editing features required within slide editing are adding textboxes and files, and formatting text colour and size (Appendix D). A markup language such as HTML (as depicted in Table 8) could be used to fulfill this requirement. This implementation would be comprehensive because it is able to accommodate any of the three sub-alternatives described above.

Table 8 This is an example of the data model table that stores the content for the slide deck. The RichText column would contain information about the content of the slide as well information about any propositions and properties contained in the slide. Other tables are not shown as more information is needed on the exact semantics of the slide feature.

Slide	RichText (e.g. XML, including proposition ID, proposition property ID, font type, size, colour, content, and html tag)
1	<prop id="123abcde-e372-a375-dfd7-abcdef123467" properties="123abcde-e372-a375-dfd7-abcdef123467" > This proposition helped us come to this conclusion.

2.8.4 Navigation within Mission

To gain a better understanding of how Missions, CCIRs, and PIRs could be accessed and modified, we iterated on a UI that presents these elements in a hierarchical fashion. A wireframe of the UI is presented in Figure 25. Appendix F shows the interface at different steps of the process we took to evolve from the tab-based navigation in the initial version of Sharik towards a hierarchical navigation scheme.

After speaking further with analysts, the client informed us that PIRs are usually established after notes and propositions have already been created. Therefore, the client also would like for PIRs to be sharable between CCIRs and notes and propositions to be sharable between PIRs. Therefore, we will add a way for the analysts to access a shared pool of unassigned PIRs, propositions, notes, and entities as we refine the design. The hierarchical structure will remain valuable – repeat PIRs will appear within different CCIRs, and repeat propositions, notes, and entities will appear within in different PIRs.

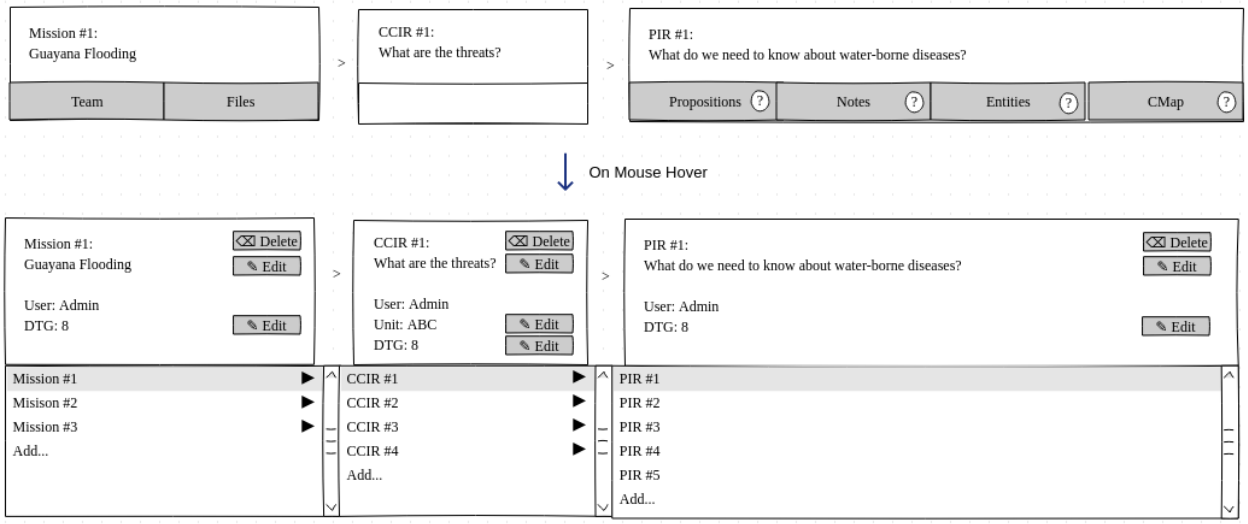


Figure 25 A hierarchical navigation UI enables an analyst to see which Mission, CCIR, and PIR is active. Tabs for changing the main view are associated with their respective level within the hierarchy. The content of the Missions, CCIRs and PIRs can be changed by hovering over the respective block and clicking edit, then modifying inline. Mission, CCIR and PIR fields (e.g. Unit, DTG) subject to change as data model is refined.

2.8.5 Evaluation of User Interface Alternatives

As outlined in Section 1.5 Objectives, the categories for usability evaluation are learnability, operability, and aesthetics. The measurement according to the metrics we have defined will require a user evaluation on an operational product. Instead we use a form of preliminary user interface evaluation to estimate the usability of a conceptual design.

A useful preliminary evaluation is a *heuristic evaluation,* where two to four several HCI specialists evaluate how well the design adheres to interface design heuristics and principles [31]. The heuristic evaluation can be less expensive and time-consuming than a usability test, and each cycle of evaluation and redesign enhances performance by around 50 percent [31]. The heuristic evaluation first identifies the most relevant interface design principles that address the tasks the product is meant to support from the guidelines in Appendix G and the display and control principles in Appendix H. For our evaluation, we eliminated several of the heuristics because either the options were not yet detailed enough to be adequately evaluated by the heuristics, or the heuristics evidently did not allow for the options to be differentiated.

We have represented the heuristic evaluation in the form of Pugh charts, which are a multi-criteria comparison chart used to compare options relative to a baseline. Since different HCI experts will be likely to discover a different set of problems, two members with HCI experience

in our Capstone design team evaluated the designs independently. We used the Pugh charts to select preferred options among the alternatives. A Pugh chart encompassing data entry is presented in Table 9, and slide creation in Table 10. The names in the column titles correspond to the option names in Sections 2.8.1 and 2.8.2.

Table 9 Pugh Chart for selecting among data entry options. Each alternative contains two columns for each HCI expert's independent evaluation. Free-form was initially chosen as baseline, however was generally ranked lower than the other two options. To facilitate a more detailed comparison, Breadcrumb was instead chosen as baseline. "0" represents approximately equivalent to baseline; "-" represents less favourable; "+" represents more favourable.

Heuristic	Breadcrumb		Free-form		CMap	
Visibility of system status	0	0	-	0	0	+
Match between system and the real world	0	0	-	-	+	+
User control and freedom	0	0	0	+	-	+
Error prevention	0	0	-	-	-	-
Recognition rather than recall	0	0	-	-	0	-
Shortcuts and accelerators	0	0	0	-	-	-
Aesthetic & minimalist design	0	0	+	+	0	-
Help users recognize, diagnose, and recover from errors (e.g. typo)	0	0	-	-	-	-
Make designs legible	0	0	-	0	+	-
Redundancy gain	0	0	0	-	+	+
Replace memory with visual information: knowledge in the world	0	0	-	-	0	+
Principle of predictive aiding	0	0	-	-	-	-
Total	**0**	**0**	**-7**	**-6**	**-2**	**-2**

Table 10 Pugh Chart for selecting among slide creation options. Each alternative contains two columns for each HCI expert's independent evaluation. "0" represents approximately equivalent to baseline; "-" represents less favourable; "+" represents more favourable.

Heuristic	Custom In-Browser		PPT+Lock		PPT+OneDrive	
Visibility of system status (e.g. real-time)	0	0	-	-	+	+
User control and freedom	0	0	+	+	+	+
Recognition rather than recall	0	0	-	-	-	-
Shortcuts and accelerators	0	0	+	+	+	+
Aesthetic & minimalist design	0	0	-	-	-	-
Principle of pictorial realism (symbolic vs actual slide design)	0	0	+	-	+	+
Principle of consistency	0	0	-	+	-	+
Total	**0**	**0**	**-1**	**-1**	**+1**	**+3**

As can be seen in Table 9, both the Breadcrumb option and CMap option perform better against the chosen metrics, with the Breadcrumb option having the slight edge. The client has also voiced appreciation for the Breadcrumb option. Considering this, we will proceed to refine the Breadcrumb option for inclusion in the solution. If time permits, the CMap option can be also included, which would increase the Redundancy Gain of the solution, in line with the Thirteen Principles of Display Design (Appendix H).

Table 10 suggests that all three options are similar in performance against the metrics. However, the PPT+Lock option does poorly in "Visibility of system status", which is a key criterion for an effective real-time collaborative tool. Therefore, we will not explore this option any further.

After the initial divergence cycle for slide creation, the client imposed the additional constraint that hosting of slides not be done on an external server for security reasons (Appendix D). Therefore, the use of OneDrive and other external hosting services has been eliminated by this constraint. However, an open-source self-hosted cloud solution remains a feasible option. This option will be further considered along with the Custom In-Browser option, for which the client has stated a preference (Appendix D). When re-evaluating these options, risk will have to be a consideration due to the complexity of implementing real-time slide editing functionality from scratch.

In a latter phase of the project, a full heuristic evaluation will be conducted on the detailed designs that will be created for the alternatives chosen above, in which potential usability problems will identified when the design violates one or more of the heuristics.

3 Project Management Plan

The design team will continue meeting with the client and supervisor in regular weekly meetings as deemed necessary. We structure our deliverables into bi-weekly deadlines.

For the rest of the project, the team will split into two autonomous teams. The front-end team will focus on UI and CSS frameworks. The back-end team will focus on data models, and backend logic. A Gantt chart is presented on the next page detailing this.

The project faces 2 major deadlines, both of which have certain expectations. Through client consultations the engineering team has identified that data-entry, the CMap, and slide generation are considered the core features of the project. These core features are placed ahead of the other features which were stated in the constraints. The two dates of interest which factor in the core components needed are:

- **Week of February 5th, 2018**: Design critique of engineering team's product so far. Should have several core components ready for demonstration (e.g. note entry, proposition entry, CMap, slide generation)
- **March 1st – March 15th, 2018**: Expected client focus group trial on engineering team's version of Sharik. Important client date for project exposure to other DRDC members and decision makers. Should have a stable software deployment hosted for DRDC to use during trial. Should have all core components (data-entry, CMap, slide generation).

GANTT CHART

Resources Legend:
Front-end: Matthew and Minna. Back-end: Michael and Alexei.
Tooling: Michael.

Grouping	Activities	Activity Types	Nov 25	Dec 07	Dec 21	Jan 04	Jan 18	Feb 01	Feb 04	Feb 15	Mar 01	Mar 15	Mar 22	Mar 26
			Weeks of											
			Exams begin ↓			Design Critique due ↓					Final Report due ↓		Showcase ↓	
Initial research & development	Perform heuristic evaluation on low-fidelity prototypes for data entry, and iterate on design	Front-end												
	Research CMap libraries and salvage existing code as much as possible, and adjust workplan	Front-end												
	Learn Spring and prototype back-end JSON endpoints	Back-end												
	Set up MySQL and create test tables	Back-end												
	Host production website on public cloud provider (e.g. Amazon Web Services)	Tooling												
	Set up development environment (e.g. Github, Jenkins, etc.)	Tooling												
	Determine if SPA architecture is viable with above prototypes	Logistics												
Intelligence lifcycle features & user management	Learn Vue.js and for prototyping front-end application	Front-end												
	Complete basic login and ICP element functionality (create users, missions, CCIRs, PIRs, etc.)													
	Create data models and endpoints for Missions, CCIRs, PIRs, IRs, and users	Back-end												
	Create views for all the intelligence items and user sign in page and creation	Front-end												
	Prototype third-party hosted cloud slide solutions and decide whether home-grown solution is needed	Back-end												
Note data-entry & evaluation of previous features	Design low-fidelity prototypes for slide creation, and iterate on design	All												
	Complete notes functionality including file and URL associativity (should have UI to interface with files hosted on the server)													
	Create file upload endpoints and file management	Back-end												
	Create note tables and related association tables (urls, tags etc)	Back-end												
	Create note user interface to enter data	Front-end												
	Evaluate progress and quality of product with objective metrics	Front-end												
CMap & Propositions	Perform user studies on all low-fidelity protoypes, and make adjustments as needed	Front-end												
	Complete minimal CMap view (display propositions)	Front-end												
	Complete proposition data-entry functionality													
	Create data models and endpoints for proposition related tables (files, urls etc)	Back-end												
	Design user interface for proposition entry (breadcrumb or other)	Front-end												
CMap Wrap Up & Slide Dissemination feature	Finish CMap functionality (able to filter on certain attributes)													
	Add endpoint support for filtering	Back-end												
	Add UI components to support filtering	Front-end												
	Refine design according to objective measurements	All												
	Finish slide functionality (export PIR notes and propositions into slides), granularity dependent on slide alternative chosen	All												
Focus group trial & Final Report Preparation	Prepare for focus group evaluations (have hosted stable solution accessible by DRDC)	Tooling												
	Start final report	Logistics												
Final Report Deliverable & Showcase Preparation	Deliver final report	Logistics												
	Prepare for showcase on March 26	Logistics												

4 Conclusion

In summary, DRDC is looking to create a tool to support the collaborative sensemaking of intelligence analysts. A user study conducted on an initial prototype of such a tool revealed aspects for improvement. In line with these aspects, we have proposed a solution that focuses on allowing rapid entry of intelligence data into the system, the visualization of intelligence data, and the creation of presentation slides to facilitate its dissemination.

The proposed solution encompasses the following components:

- A single page web architecture which involves a browser, web server and database
- A MySQL database to store Sharik's data along with a file system to store uploaded digital assets. The data models will incorporate client feedback as the project progresses.
- A Vue.js framework for rendering the user interface on the browser (runs on the browser) that the analyst uses. Vue.js is based on JavaScript. Twitter's Bootstrap CSS framework will be used alongside Vue.js.
- A Spring Java framework for the back-end software processes (this will be running on the web server).
- The breadcrumb option for data entry UI.
- The PPT+OneDrive and custom in-browser options for collaborative slide creation UI. We will refrain from using external hosts for slide decks due to security concerns as expressed by the client.

It is noted that the core features of the project which are data-entry, the CMap, and slide generation must be stable by the planned focus group trial by early March 2018.

To augment the central goals of the project, the design aims to support many simultaneous analysts and will strive to implement supplementary feature requests if time permits. Once the design proposal is reviewed by the project supervisor and the client, the design team will begin the implementation stage of the project.

5 References

[1] S. Ghajar-Khosravi and P. Kwantes, "Sharik 2.0: The Design and Development of a Web-Based Tool to Support Collaborative Sensemaking," DRDC-RDDC, Toronto, ON, Aug. 2017.

[2] M. Townsend, "How a crippling shortage of analysts let the London Bridge attackers through | UK news | The Guardian," 11 June 2017. [Online]. Available: https://www.theguardian.com/uk-news/2017/jun/10/london-bridge-attackers-intelligence-overload. [Accessed 12 October 2017].

[3] ISO, "ISO/IEC 25010:2011 Systems and software engineering: Systems and software Quality Requirements and Evaluation (SQuaRE): System and software quality models," ISO, Geneva, 2011.

[4] ISO, "ISO/IEC 25023:2016 Systems and software engineering: Systems and software Quality Requirements and Evaluation (SQuaRE): Measurement of system and software product quality," ISO, Geneva, 2016.

[5] S. Sinofsky, "Scaling to different screens – Building Windows 8," 21 March 2012. [Online]. Available: https://blogs.msdn.microsoft.com/b8/2012/03/21/scaling-to-different-screens/. [Accessed 6 October 2017].

[6] Microsoft, "Chapter 20: Choosing an Application Type," Oct 2009. [Online]. Available: https://msdn.microsoft.com/en-us/library/ee658104.aspx. [Accessed 23 Nov 2017].

[7] G. S. Palani, "Cloud or desktop? Compare and contrast applications," 18 April 2011. [Online]. Available: https://www.ibm.com/developerworks/cloud/library/cl-cloudordesktop/index.html. [Accessed 22 Nov 2017].

[8] Stackify, "What is N-Tier Architecture? How It Works, Examples, Tutorials, and More," 21 Aug 2017. [Online]. Available: https://stackify.com/n-tier-architecture/. [Accessed 22 Nov 2017].

[9] M. Fowler, Patterns of Enterprise Application Architecture, Addison-Wesley, 2002.

[10] M. Wasson, "Single-Page Applications: Build Modern, Responsive Web Apps with ASP.NET," Nov 2013. [Online]. Available: https://msdn.microsoft.com/en-us/magazine/dn463786.aspx. [Accessed 22 Nov 2017].

[11] A. Zolciak, "SPA vs MPA. What Are the Pros and Cons?," 19 Aug 2017. [Online]. Available: https://insanelab.com/blog/web-development/spa-vs-mpa-single-multi-page-application-pros-cons/. [Accessed 22 Nov 2017].

[12] BBVAOPEN4U, "REST API: What is it, and what are its advantages in project development?," 23 March 2016. [Online]. Available: https://bbvaopen4u.com/en/actualidad/rest-api-what-it-and-what-are-its-advantages-project-development. [Accessed 22 Nov 2017].

[13] S. Shimanovsky, "Multi page web applications vs. single page web applications," 9 July 2015. [Online]. Available: http://www.eikospartners.com/blog/multi-page-web-applications-vs.-single-page-web-applications. [Accessed 22 Nov 2017].

[14] M. Wasson and R.-B. Lee, "Choose the right data store," 9 Aug 2017. [Online]. Available: https://docs.microsoft.com/en-us/azure/architecture/guide/technology-choices/data-store-overview. [Accessed 22 Nov 2017].

[15] J. Homan, "Relational vs. non-relational databases: Which one is right for you?," 6 April 2014. [Online]. Available: https://www.pluralsight.com/blog/software-development/relational-non-relational-databases. [Accessed 22 Nov 2017].

[16] M. Wasson, "Criteria for choosing a data store," 9 Aug 2017. [Online]. Available: https://docs.microsoft.com/en-us/azure/architecture/guide/technology-choices/data-store-comparison. [Accessed 22 Nov 2017].

[17] K. Rangadurai, "Git Internals - How Git works," 22 Jan 2015. [Online]. Available: https://www.linkedin.com/pulse/git-internals-how-works-kaushik-rangadurai/. [Accessed 22 Nov 2017].

[18] DB-Engines, "Popularity of open source DBMS versus commercial DBMS," Nov 2017. [Online]. Available: https://db-engines.com/en/ranking_osvsc. [Accessed 22 Nov 2017].

[19] DB-Engines, "DB-Engines Ranking of Relational DBMS," Nov 2017. [Online]. Available: https://db-engines.com/en/ranking/relational+dbms. [Accessed 22 Nov 2017].

[20] DB-Engines, "Method of calculating the scores of the DB-Engines Ranking," Nov 2017. [Online]. Available: https://db-engines.com/en/ranking_definition. [Accessed 22 Nov 2017].

[21] G. Bakradze, "Choosing the Best Front-end Framework," Toptal, 14 September 2017. [Online]. Available: https://www.toptal.com/javascript/choosing-best-front-end-framework. [Accessed 18 November 2017].

[22] S. Krause, "Interactive Results," 29 May 2017. [Online]. Available: http://www.stefankrause.net/js-frameworks-benchmark6/webdriver-ts-results/table.html. [Accessed 19 November 2017].

[23] S. Greif, "Front-end Frameworks," 11 October 2016. [Online]. Available: http://stateofjs.com/2016/frontend/. [Accessed 18 November 2017].

[24] S. Greif, "The State Of JS 2017 Results Preview," 5 October 2017. [Online]. Available: https://medium.com/@sachagreif/the-state-of-js-2017-results-preview-acbd2885da7f. [Accessed 19 November 2017].

[25] "Comparison with Other Frameworks — Vue.js," 12 October 2017. [Online]. Available: https://vuejs.org/v2/guide/comparison.html. [Accessed 19 November 2017].

[26] HotFrameworks, "Web framework rankings | HotFrameworks," 21 October 2016. [Online]. Available: https://hotframeworks.com/. [Accessed 18 November 2017].

[27] D. Gilling, "Popular languages for building RESTful APIs based on our usage data Moesif's Musings on Software," Moesif, 12 August 2017. [Online]. Available: https://www.moesif.com/blog/engineering/api-analytics/Popular-Languages-For-Building-RESTful-APIs-Based-On-Our-Usage-Data/. [Accessed 18 November 2017].

[28] Mozilla, "Server-side web frameworks - Learn web development | MDN," 1 September 2017. [Online]. Available: https://developer.mozilla.org/en-US/docs/Learn/Server-side/First_steps/Web_frameworks. [Accessed 18 November 2017].

[29] Netcraft, "October 2017 Web Server Survey," 26 October 2017. [Online]. Available: https://news.netcraft.com/archives/2017/10/26/october-2017-web-server-survey-13.html. [Accessed 21 November 2017].

[30] Slashdot, "Survey Finds Most Popular Linux Laptop Distros: Ubuntu and Arch - Slashdot," 9 July 2017. [Online]. Available: https://linux.slashdot.org/story/17/07/08/2346248/survey-finds-most-popular-linux-laptop-distros-ubuntu-and-arch. [Accessed 19 November 2017].

[31] C. Wickens, J. Lee, Y. Liu and S. Gordan Becker, An Introduction to Human Factors Engineering 2nd edition, Pearson Education, 2004, p. Chapter 8.

[32] S. Ghajar-Khosravi, S. Gibbon and P. Kwantes, "Usability and Usefulness Evaluation of Sharik 2.0," Defence Research and Development Canada, Toronto, 2017.

[33] J. Nielson and L. Mack, "Heuristic evaluation," *Usability Inspection Methods,* 1994b.

[34] "7 Best Frameworks For Web Development in 2017," 22 May 2017. [Online]. Available: https://gearheart.io/blog/7-best-frameworks-for-web-development-in-2017/. [Accessed 15 November 2017].

[35] T. Gray, "What's The Best Restful Web API Framework? – Part 5 | OptimalBI," OptimalBI, 28 November 2016. [Online]. Available: https://optimalbi.com/blog/2016/11/28/whats-the-best-restful-web-api-framework-part-5/. [Accessed 17 November 2017].

Appendix A
Glossary

Detailed definitions of terminology

Concept Map: A visual map of entities with relationships drawn between them. Used for visualizing propositions.
Failure: The inability for any Sharik function to run as intended.
Note: Free-form textual information with supplementary files/URLs supporting credibility. Also has other attributes such as reliability and credibility scores as defined in Appendix J.
Proposition: A relationship between entities (e.g. people, places), along with additional details represented as properties (Appendix J).

Definitions of Design for Excellence (DfX) and Sub-categories for DfX [3]

1. Design for Performance Efficiency: *performance relative to the amount of resources used under stated conditions*
    a. Time Behaviour: *degree to which the response and processing times and throughput rates of a product or system, when performing its functions, meet requirements*
2. Design for Usability: *the degree to which a product or system can be used by specified users to achieve specified goals with effectiveness, efficiency and satisfaction in a specified context of use*
    a. Learnability: *degree to which a product or system can be used by specified users to achieve specified goals of learning to use the product or system with effectiveness, efficiency, freedom from risk and satisfaction in a specified context of use*
    b. Operability: *degree to which a product or system has attributes that make it easy to operate and control*
    c. User interface aesthetics: *degree to which a user interface enables pleasing and satisfying interaction for the user*
3. Design for Maintainability: *degree of effectiveness and efficiency with which a product or system can be modified by the intended maintainers*
    a. Testability: *degree of effectiveness and efficiency with which test criteria can be established for a system, product or component and tests can be performed to determine whether those criteria have been met*
4. Design for Reliability: *degree to which a system, product or component performs specified functions under specified conditions for a specified period of time*
    a. Maturity: *degree to which a system, product or component meets needs for reliability under normal operation*
    b. Recoverability: *degree to which, in the event of an interruption or a failure, a product or system can recover the data directly affected and re-establish the desired state of the system*

5. Design for Functional Suitability: *degree to which a product or system provides functions that meet stated and implied needs when used under specified conditions*
    a. Functional Completeness: *degree to which the set of functions covers all the specified tasks and user objectives*

Appendix B
Pair-wise Comparison Chart

Pair-wise Comparison Chart			Design for Performance Efficiency		Design for Usability					Design for Maintainability	Design for Reliability		Design for Functional Suitability		
			Time Behavior		Learnability			Operability	User Interface Aesthetics	Testability	Maturity	Recoverability	Functional Completeness		
			Mean throughput of task	Mean response time of load	User guidance completeness	Entry field placeholders	Error messages understandability	Outcome message clarity	Appearance aesthetics of user interfaces	Test function completeness	Failure rate	Mean recovery time	Functional coverage of optional features	Score	Category Score (Average)
Design for Performance Efficiency	Time Behavior	Mean throughput of task		1	1	1	1	1	1	1	1	1	1	10	9.5
		Mean response time of load	0		1	1	1	1	1	1	1	1	1	9	
Design for Usability	Learnability	User guidance completeness	0	0		0	0	1	1	1	1	1	1	6	5.8
		Entry field placeholders	0	0	1		0	1	1	1	1	1	1	7	
		Error messages understandability	0	0	1	1		1	1	1	1	1	1	8	
	Operability	Outcome message clarity	0	0	0	0	0		1	1	1	1	1	5	
	User Interface Aesthetics	Appearance aesthetics of user interfaces	0	0	0	0	0	0		1	0	1	1	3	
Design for Maintainability	Testability	Test function completeness	0	0	0	0	0	0	0		1	1	1	3	3
Design for Reliability	Maturity	Failure rate	0	0	0	0	0	0	1	0		0	1	2	2
	Recoverability	Mean recovery time	0	0	0	0	0	0	0	0	1		1	2	
Design for Functional Suitability	Functional Completeness	Functional coverage of optional features	0	0	0	0	0	0	0	0	0	0		0	0

Figure 26 Comparison of the ten objectives

Note:

1. The binary decisions are based on conversations with the client and the client document and the final objective list ordering has been confirmed with the client through email communication (see Appendix C).
2. To add more clarity, levels are identified by different shades.

Appendix C
Email confirmation of objective list priority ordering

From: Ghajar-Khosravi, Shadi
Sent: Monday, October 16, 12:00 PM
Subject: RE: Confirmation of Objectives Priority
To: Alexei Zenin

Hi Alexei,

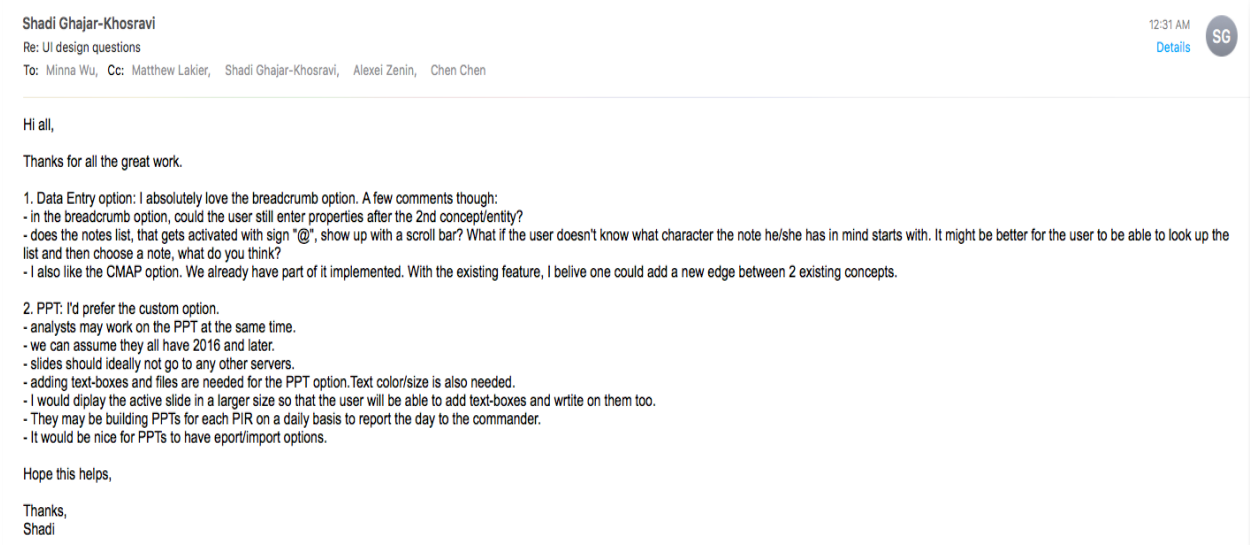
I agree with and confirm the list and order of objectives listed under section 1.3. Thank you.

Shadi

From: Alexei Zenin [mailto:alexei.zenin@mail.utoronto.ca]
Sent: October-16-17 11:54 AM
To: Ghajar-Khosravi, Shadi; ShadiGhajar-Khosravi
Subject: Confirmation of Objectives Priority

Hi Shadi,
The team would want to have you confirm the priority of the objectives (the order they are listed in) in the document we sent you and if they best reflect what would be needed for the future of Sharik.
Alexei

Appendix D
Email confirmation of client preferences for alternative



Shadi Ghajar-Khosravi

12:31 AM

Details

SG

Re: UI design questions

To: Minna Wu, Cc: Matthew Lakier, Shadi Ghajar-Khosravi, Alexei Zenin, Chen Chen

Hi all,

Thanks for all the great work.

1. Data Entry option: I absolutely love the breadcrumb option. A few comments though:
- in the breadcrumb option, could the user still enter properties after the 2nd concept/entity?
- does the notes list, that gets activated with sign "@", show up with a scroll bar? What if the user doesn't know what character the note he/she has in mind starts with. It might be better for the user to be able to look up the list and then choose a note, what do you think?
- I also like the CMAP option. We already have part of it implemented. With the existing feature, I belive one could add a new edge between 2 existing concepts.

2. PPT: I'd prefer the custom option.
- analysts may work on the PPT at the same time.
- we can assume they all have 2016 and later.
- slides should ideally not go to any other servers.
- adding text-boxes and files are needed for the PPT option.Text color/size is also needed.
- I would diplay the active slide in a larger size so that the user will be able to add text-boxes and wrtite on them too.
- They may be building PPTs for each PIR on a daily basis to report the day to the commander.
- It would be nice for PPTs to have eport/import options.

Hope this helps,

Thanks,
Shadi

Appendix E
Example webpage layout for UI

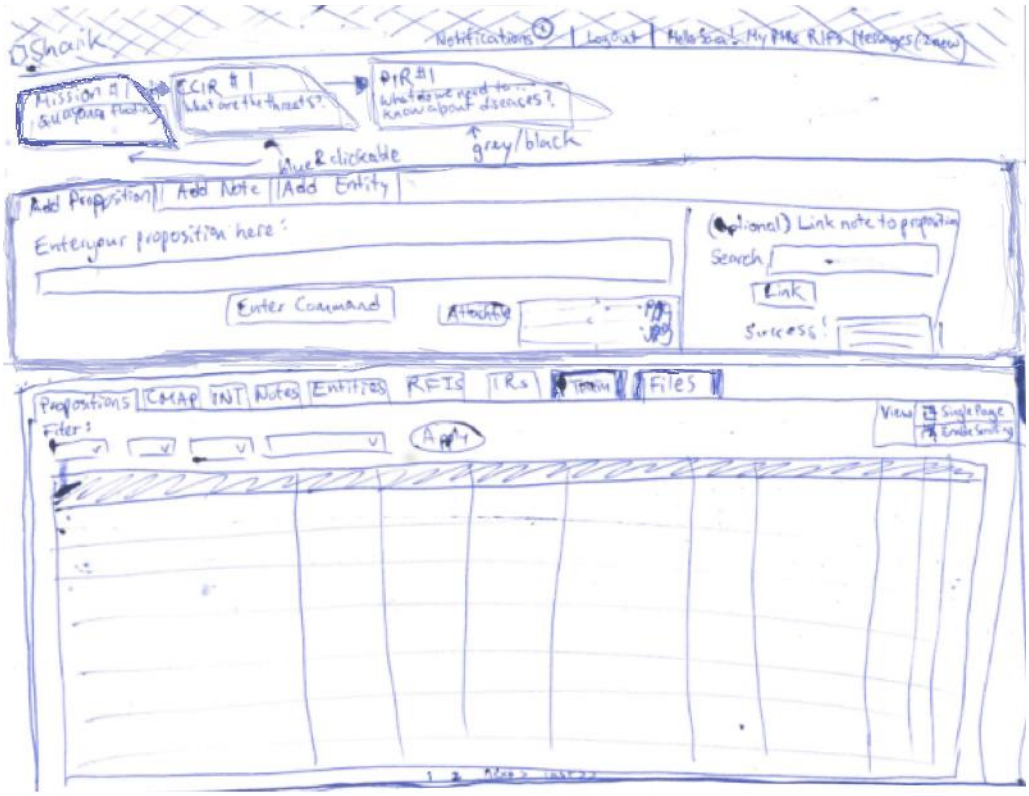


Figure 27 One possible layout for the UI layout of the solution. Navigation controls are at the top of the page. Data entry controls are in the middle of the page. Information about the currently selected tab is at the bottom of the page.

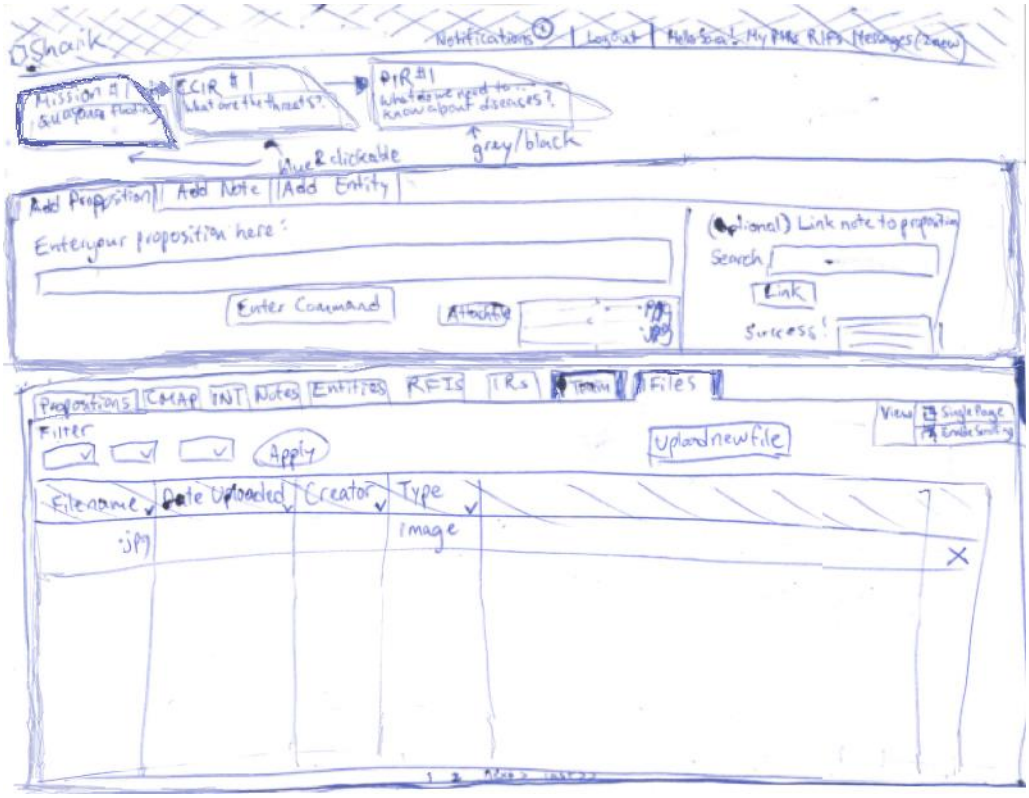


Figure 28 The contents of the file tab displayed in the centre of the page. Details such as filename, file upload date, creator, and file type are presented.

Appendix F
Design of Hierarchical Mission Navigation

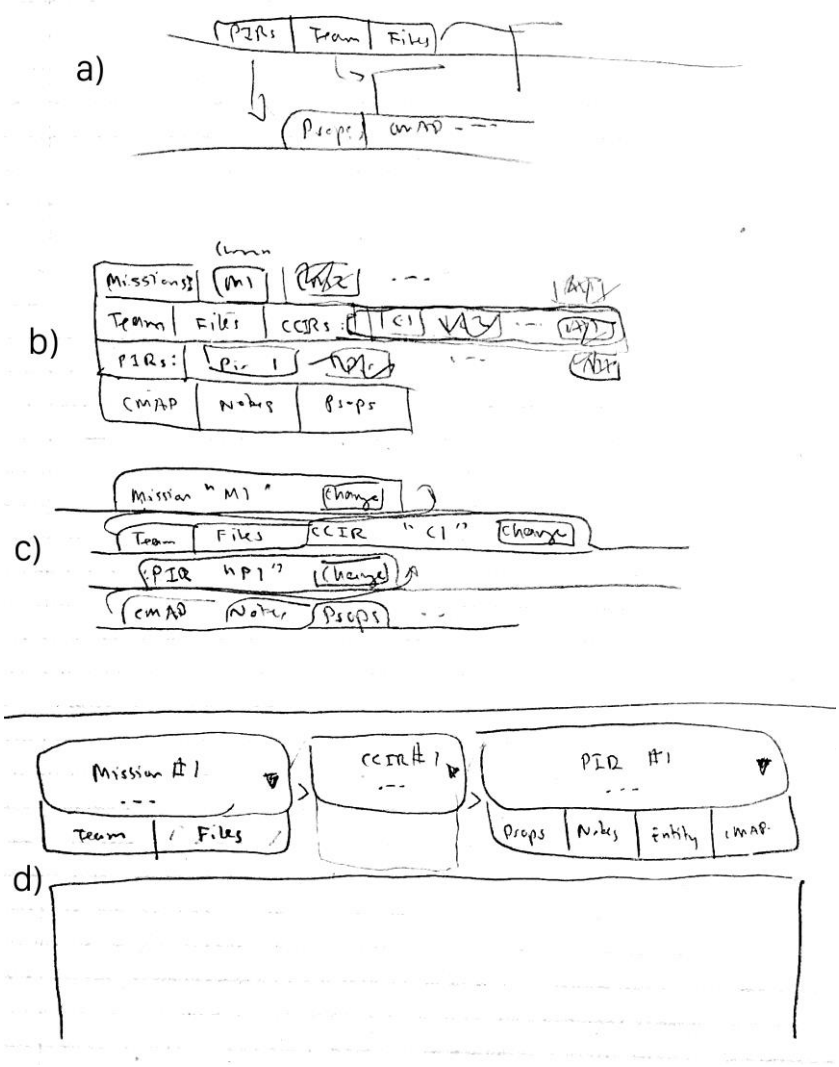


Figure 29 Iterations of layouts for the hierarchical navigation of missions. (a): The teams and files tabs are associated with the CCIR display, and PIR tabs are nested one level below. (b, c): Various ways of presenting the CCIR and PIR tabs along with buttons to change the active Mission, CCIR, and PIR. (d): Rearranged version of (c) that presents Missions, CCIR, and PIR as dropdown boxes rather than tabs.

Appendix G
Nielsen's Ten Heuristics for User Interface Design

The below set of heuristics were derived by Jakob Nielsen, a prominent figure in user-centered design, from a factor analysis of 249 usability problems to maximum explanatory power [33].

1. Visibility of system status: The system should always keep users informed about what is going on, through appropriate and timely feedback, especially when the system takes control, or an action is taken.

2. Match between system and the real world: The system should speak the users' language, rather than in system-oriented terms. It should follow real-world conventions, making information appear in a natural and logical order.

3. User control and freedom: User should be able to undo and redo, stop browser processing at any time, and go back to a previous step.

4. Consistency and standards: Users should not have to wonder whether different words, actions, or commands mean the same thing.

5. Error prevention: Careful design prevents errors from occurring in the first place. If this is not possible, the design should minimize the negative consequences of errors and help users recover from their errors. A good error message explicitly indicates a problem, precisely describes that problem, and offers constructive advice in human-readable language that doesn't blame the user.

6. Recognition rather than recall: See and point, not remember and type

7. Shortcuts and accelerators: Provide accelerators for experts that do not get in the way of novices

8. Aesthetic & minimalist design: Every extra unit of information in a dialogue competes with the relevant units of information and diminishes their relative visibility.

9. **Help users recognize, diagnose, and recover from errors:** Error messages should be expressed in plain language (no codes), precisely indicate the problem, and constructively suggest a solution.

10. **Help and documentation:** Help or documentation should be easy to search, be task-specific, and list steps to be carried out.

Appendix H
Thirteen Principles of Display Design

The following thirteen heuristics are compiled in the An Introduction to Human Factors Engineering textbook. Some of the following thirteen principles conflict, so judgement is required to assess when exceptions are beneficial and should occur [31].

Perceptual Design Principles:

1. **Make designs legible**: Avoid issues with contrast, visual angle, illumination, noise, and masking.
2. **Avoid absolute judgement limits**: Avoid having the user distinguish between more than five different levels of a variable.
3. **Top-down processing**: People perceive based on their experience. If a signal is presented contrary to expectations (e.g. a warning message for an unlikely event), it will likely be ignored unless *more* evidence of that signal is presented in the *immediate* context.
4. **Redundancy gain:** A message is more likely to be interpreted correctly when the same message is expressed more than once, especially in alternative physical forms (e.g. print and pictures, color and shape).
5. **Discriminability: Similarity causes confusion: Use discriminable elements:** When confusion could be serious, unnecessary similar features should be deleted and dissimilar features should be highlighted.

Mental Model Principles:

6. **Principle of pictorial realism:** A display should look like the variable it represents.
7. **Principle of the moving part:** Moving elements of a display should move in a spatial pattern and direction that is compatible with the user's mental model of how the represented element moves in the physical system.

Principles based on attention:

8. **Minimizing information access cost:** Keep frequently accessed sources close to each other to reduce the user's need to "move" their selective attention.
9. **Proximity compatibility principle:** Keep information sources that need to be mentally integrated for a task close or linked by a common color or line (but not cluttered or overlapping).
10. **Principle of multiple resources**: If possible, divide large amounts of information across resources (e.g. present visual and auditory information concurrently rather than all visually or all auditory).

Memory Principles:

11. **Replace memory with visual information: knowledge in the world:** Users should not be required to retain important information solely in working memory or retrieve it from long-term memory.

12. **Principle of predictive aiding:** Reduce resource-demanding cognitive tasks by replacing them with simpler reactive tasks by predicting future conditions.
13. **Principle of consistency**: Aim to design displays that are consistent with what the user has been using or is using concurrently.

Appendix I
Relational Model Primer

The relational model organizes information into tables. Each table contains attributes (known as columns). Records are stored as rows within the table which provide certain values for the defined columns. For instance, say we had a table named Artists which stores musical artists. It has an ArtistID (to uniquely identify the row or artist by a uniquely assigned number), a name column to store the artist's name, and a date of birth column to store when the artist was born. Below is an example of the schema for the table; a schema defines the structure of a table.

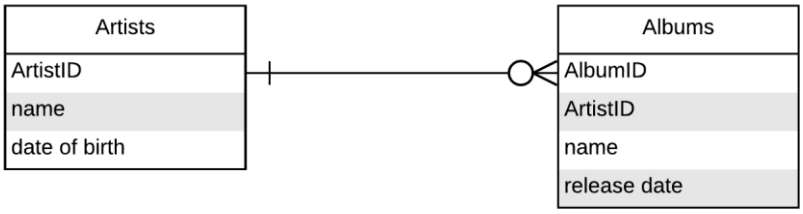
Artists Table
ArtistID
name
Date of birth (DOB)

A realization of this table where data is populated or inserted into this schema is when values are provided for each of the columns defined above and inserted as a row. An example is given below. Each row is sometimes called a record or instance of an artist (extracting the row will produce the set or tuple of columns that represent an artist in the data model).

ArtistID	name	DOB
1	Michael Jackson	August 29, 1958
2	Malcolm Mitchell Young	January 6, 1953
3	Madonna Louise Ciccone	August 16, 1958

In the above example if we were to query for the artist with id equal to 1 we would get the first row as the result (Michael Jackson). The notion of a column or set of columns that identify the row uniquely are called a primary key. In this case we have defined that we wish to have the ArtistID column as the primary key.

Primary keys are used to link data to other tables without repeating the same information in another table (normalization, the production of schemas which achieve a mathematical form which reduces data redundancy or achieves other characteristics). If we now add an Albums table, we see that we must now link an artist with many albums (due to the domain requirements, artists can have zero or more albums attributed to them). This is illustrated below.



In this schema, the arrow signifies that for each artist record that it can be associated with many or zero albums (the circle and trident). The vertical line on the arrow near the Artists table signifies for each album there may only be one artist, a constraint we wish to enforce. This is done by adding a foreign key column to Albums, namely ArtistID, which refers to the primary key in the Artists table ArtistID. Foreign keys in general are primary keys from another table to which you would like to associate with and be able to join two tables later when running queries. A realization is shown below with mock data.

Artists Table

ArtistID	name	DOB
1	Michael Jackson	August 29, 1958
2	Malcolm Mitchell Young	January 6, 1953
3	Madonna Louise Ciccone	August 16, 1958

Albums Table

AlbumID	ArtistID	name	Release date
1	1	Thriller	November 14, 1983
2	1	Bad	August 31, 1987
3	2	Back in Black	25 July 1980

As seen above the Albums table links back to the artist who made the album by including the artist id in the foreign key column (ArtistID). For example, if we wanted to retrieve all albums for Michael Jackson we would simply request to join both tables (combine columns making a row longer) by joining rows only where the ArtistID in the Artists table is equal to 1 (Michael Jackson's assigned id in our case) and where the rows in the Albums table has its ArtistID also equal to one. This operation runs for each row in each table, thus it will produce up to the number of records in the albums table. We would get "x" number of rows for "x" number of albums for a single artist.

The result of this query or request results in a new table (not actually defined in the model but created in a sense on the fly).

Resulting schema of the query

Artists.ArtistID	Artists.name	Artists.DOB	Albums.AlbumID	Albums.ArtistID	Albums.name	Albums.Release date
1	Michael Jackson	August 29, 1958	1	1	Thriller	November 14, 1983
1	Michael Jackson	August 29, 1958	2	1	Bad	August 31, 1987

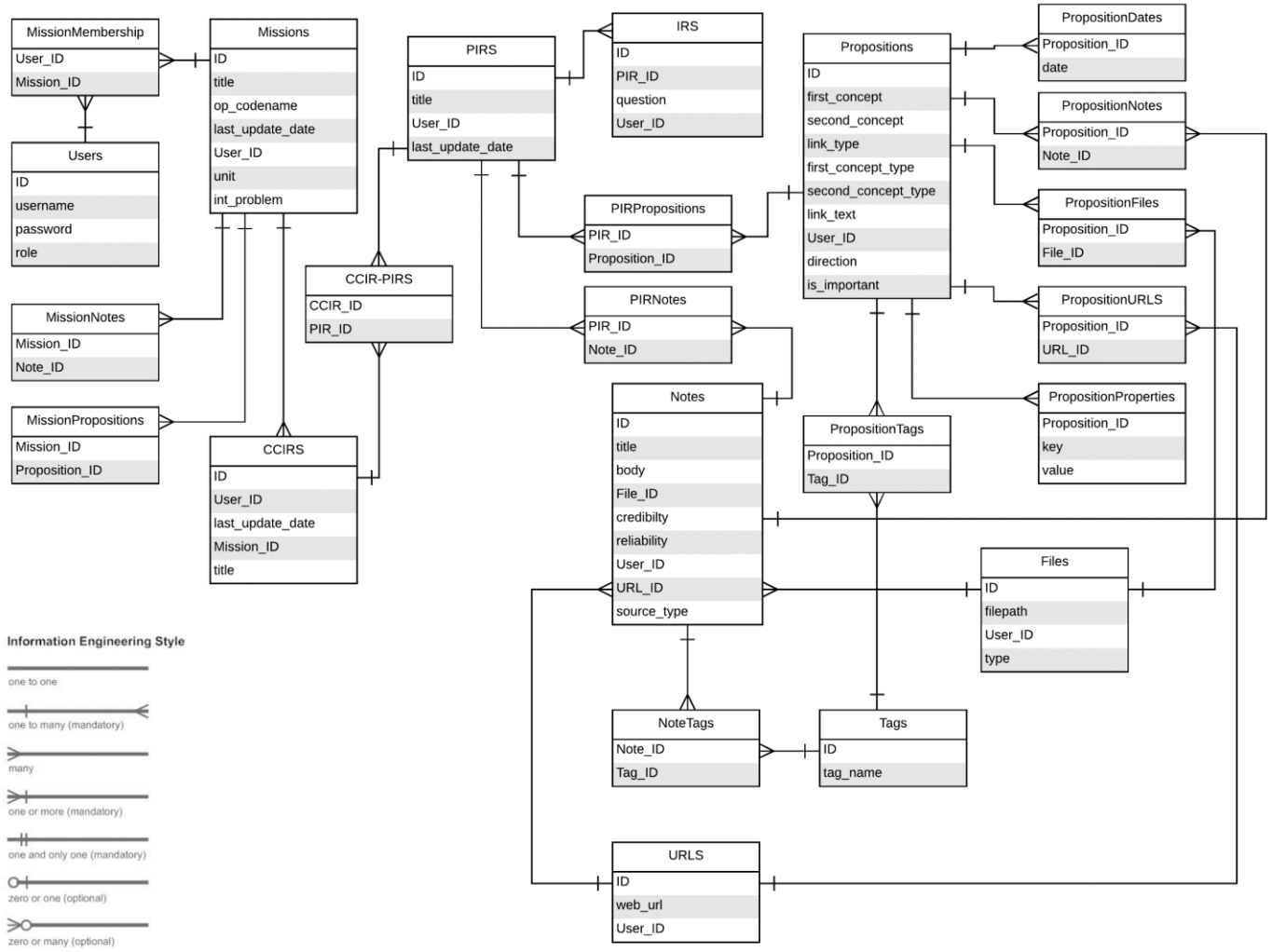
As seen above, two rows were returned as expected since Michael Jackson had only two associated albums (Thriller and Bad) saved in the model. We can see that the columns between the two tables merged and are available for inspection for each row result.

It can be noted that the model defined above allows for an artist record to exist but have no associated albums yet (maybe they are a new artist, or we have not uploaded all their data yet) as is the case for Madonna (there are no album records with Madonna's artist id).

By storing data in rows and creating tables which provide logical separation of entities with logical associations between each other, the data model or relational schema provides strict constraints on how data can be inserted and modified. This type of modelling is well suited to situations where the data is well known and needs to be constrained and follow a specific structure.

Appendix J
Sharik Model In-Depth

The following appendix will describe all columns within each table defined in section 2.2. The primary keys are highlighted in each table (if more than one is highlighted than that set of columns is considered the composite primary key – a row can only exist if all values in the set are unique). The set of table descriptions pertain to the figure below, placed in the appendix for convenience. Note that attributes with each word being capitalized indicates a foreign key.



Users Table

Stores user accounts within the Sharik system.

Column	Description	Data Type
ID	Uniquely identifying number	Number
username	The username the user will be known as and sign in with	Alphanumeric
password	The cryptographic hash of the password. Prevents hackers from obtaining the clear text password if the database is comprised without additional effort	Alphanumeric
role	The role of analyst within the Sharik environment. Currently this can be set to one of two values: “Admin” and “Analyst”. The specific rights they have within the application will be determined by the client later (e.g. only admins can create new user accounts)	Alphanumeric

Missions Table

Stores the missions added to the Sharik system.

Column	Description	Data Type
ID	Uniquely identifying number	Number
title	The title of the mission	Alphanumeric
op_codename	The operation codename for the mission	Alphanumeric
last_update_date	The last date the mission was updated in its attributes	Date
User_ID	Foreign key to the Users table. Indicates who created the mission	Number
Unit	The intelligence unit the mission is part of	Alphanumeric
int_problem	The intelligence problem in words that the mission is addressing	Alphanumeric

CCIR Table

Stores the Critical Commanders Information Requirements.

Column	Description	Data Type
ID	Uniquely identifying number	Number
title	The title of the CCIR	Alphanumeric
Mission_ID	Foreign key to the Missions table. The mission the CCIR is part of	Number
last_update_date	The last date the CCIR was updated in its attributes	Date
User_ID	Foreign key to the Users table. Indicates who created the CCIR	Number

PIR Table

Stores the Priority Information Requirements.

Column	Description	Data Type
ID	Uniquely identifying number	Number
title	The title of the PIR	Alphanumeric
last_update_date	The last date the PIR was updated in its attributes	Date
User_ID	Foreign key to the Users table. Indicates who created the PIR	Number

IRS Table

Stores the Information Requirements.

Column	Description	Data Type
ID	Uniquely identifying number	Number
question	The question of the IR	Alphanumeric
PIR_ID	Foreign key to the PIR table. Indicates which PIR the IR is for	Number
User_ID	Foreign key to the Users table. Indicates who created the PIR	Number

Notes Table

Stores the Sharik Note. Contains references to other resources such as tags, files and URLs.

Column	Description	Data Type
ID	Uniquely identifying number	Number
title	The title of the intelligence note	Alphanumeric
body	The main content of the intelligence note goes here	Alphanumeric
File_ID	Foreign key to the File table. Associates a file to the intelligence note. Only permits one file to be associated	Number
credibility	An enumerated value based on the credibility of the primary source of information the note was generated from (as defined by the Department of Defense).	Possible values • Confirmed by other sources • Probably true • Possibly true • Doubtful • Improbable • Not Credible
reliability	An enumerated value based on the reliability of the primary source of information the note was generated from (as defined by the Department of Defense)	Possible values • Completely reliable • Usually reliable • Fairly reliable • Not usually reliable • Unreliable • Reliability cannot be judged
User_ID	Foreign key to the Users table. Indicates who created the Note	Number
URL_ID	Foreign key to the URLs table. Associates a web URL as a source for the note. Only permits one URL to be associated	Number
source_type	Indicates the type of the source of information (e.g. human, signal)	Alphanumeric

Propositions Table

Stores Sharik Propositions. Contains all necessary fields to display visually in Concept Map.

Column	Description	Data Type
ID	Uniquely identifying number	Number
first_concept	The first concept the proposition is composed of. This can be any entity such as a person, place or thing (e.g. Toronto, Jason Bourne)	Alphanumeric
second_concept	The second concept the proposition is composed of. Same as above.	Alphanumeric
link_type	This field specifies whether the relationship described in the proposition is a fact, conjecture or extract. A fact is deemed as completely true without supporting sources, a conjecture is unknown whether to be true or false and an extract is a fact backed by supporting sources. A fact may change to an extract when supporting sources are linked to the proposition and deemed satisfactory.	Possible Values • Fact • Conjecture • Extract
first_concept_type	An enumerated value which describes what the first concept type is (Event, Individual etc.).	Possible values • Individual • Place • Group • Role • Thing • Event
second_concept_type	Same as above except for the second concept	Possible values • Individual • Place • Group • Role • Thing • Event
link_text	The set of words which relate the first concept to the second (Toronto *is a* City)	Alphanumeric

User_ID	Foreign key to the Users table. Indicates who created the proposition	Number
direction	An enumerated value which describes what the direction of the relationship is for the proposition between the first and second concept (unidirectional, bidirectional).	Possible value • Unidirectional • Bidirectional
is_important	A marker indicating if the proposition is important (like the flagged feature in an email)	Boolean (True/False)

Proposition Dates Table

Stores a set of dates related to a single proposition. Each date indicates when the proposition occurred. Note cannot insert duplicate dates for the same proposition id as constrained by the composite primary key.

Column	Description	Data Type
Proposition_ID	Foreign key to the Propositions table. Indicates the proposition the date refers to.	Number
date	The date when the proposition occurred	Date

Proposition Properties

Stores a set of extra fields that can be attached to a proposition. Allows to enter free text properties not already captured by the proposition itself. A property is defined as a unique key (for the proposition) and a value (key = latitude, value = 78.44 or key = location, value = Toronto).

Column	Description	Data Type
Proposition_ID	Foreign key to the Propositions table. Indicates the proposition the property refers to	Number
key	The key of the property	Alphanumeric
value	The value of the key for the property	Alphanumeric

Files Table

Stores the file paths of files uploaded to Sharik. The actual files are stored on the local file system of the server. Can be used in notes and propositions as sources.

Column	Description	Data Type
ID	Uniquely identifying number	Number
filepath	The filepath of the uploaded file on the server (e.g. /root/file.txt)	Alphanumeric
type	The type of file uploaded	Alphanumeric
User_ID	Foreign key to the Users table. Indicates who uploaded the file	Number

URLS Table

Stores web URL sources that analysts can use in notes and propositions.

Column	Description	Data Type
ID	Uniquely identifying number	Number
web_url	The web URL of the source (e.g. http://cnn.com)	Alphanumeric
User_ID	Foreign key to the Users table. Indicates who uploaded the URL	Number

Tags Table

Stores tags used by analysts to mark data into related categories for later querying. Used to tag notes and propositions.

Column	Description	Data Type
ID	Uniquely identifying number	Number
tag_name	The name of the tag (e.g. Saudi Arabia)	Alphanumeric

The above tables summarize the main storage tables without the tables which contain the table-table associations. All association tables are structured the same way (groups of foreign keys) as with the Mission Membership being shown below.

Mission Membership Table

Stores which users are assigned to what mission. Each row indicates a membership for a user to a certain mission and a mission to a user.

Column	Description	Data Type
User_ID	Foreign key to Users table. Indicates who is part of a mission	Number
Mission_ID	Foreign key to Missions table. Indicates the mission someone is part of	Number

The rest of the association tables will be described by listing the associations they provide instead of their descriptions.

Association tables:

- The tables PropositionNotes, PropositionFiles, PropositionURLS, PropositionTags all provide the ability for many notes, files, URLs and tags to be added to a proposition
- The tables PIRPropositions allows for propositions to be added to multiple different PIRS, the same is enabled for notes through the PIRNotes table
- The NoteTags table allows the addition of multiple tags to a note
- The CCIR-PIRS table allows the addition of PIRS to many different CCIRs
- The MissionNotes and MissionPropositions allows the addition of many different propositions and notes to a mission (this allows the addition of notes and propositions to the system even if there are no PIRS yet as per client requirements). Note that in the diagram a one-to-many relationship was missing between these tables and the respective Notes and Propositions tables (one-to-many from Notes/Propositions to the associative table)

Missing Associative lines:

- The Users table was missing one-to-many relationships between itself and many other tables such as: All the ICP element tables, Notes, Propositions, Files, and URLS.

DOCUMENT CONTROL DATA

*Security markings for the title, authors, abstract and keywords must be entered when the document is sensitive

1. ORIGINATOR (Name and address of the organization preparing the document. A DRDC Centre sponsoring a contractor's report, or tasking agency, is entered in Section 8.)

University of Toronto
27 King's College Cir, Toronto, ON M5S 3H7

2a. SECURITY MARKING (Overall security marking of the document including special supplemental markings if applicable.)

CAN UNCLASSIFIED

2b. CONTROLLED GOODS

NON-CONTROLLED GOODS
DMC A

3. TITLE (The document title and sub-title as indicated on the title page.)

Sharik 3.0 Design Proposal

4. AUTHORS (Last name, followed by initials – ranks, titles, etc., not to be used)

Zenin, A.; Lakier, M.; Chen, M.; Wu, M.

5. DATE OF PUBLICATION (Month and year of publication of document.)	6a. NO. OF PAGES (Total pages, including Annexes, excluding DCD, covering and verso pages.)	6b. NO. OF REFS (Total references cited.)
November 2017	71	35

7. DOCUMENT CATEGORY (e.g., Scientific Report, Contract Report, Scientific Letter.)

Contract Report

8. SPONSORING CENTRE (The name and address of the department project office or laboratory sponsoring the research and development.)

DRDC – Toronto Research Centre
Defence Research and Development Canada
1133 Sheppard Avenue West
Toronto, Ontario M3K 2C9
Canada

9a. PROJECT OR GRANT NO. (If appropriate, the applicable research and development project or grant number under which the document was written. Please specify whether project or grant.)	9b. CONTRACT NO. (If appropriate, the applicable number under which the document was written.)
05da	

10a. DRDC PUBLICATION NUMBER (The official document number by which the document is identified by the originating activity. This number must be unique to this document.)	10b. OTHER DOCUMENT NO(s). (Any other numbers which may be assigned this document either by the originator or by the sponsor.)
DRDC-RDDC-2018-C212	

11a. FUTURE DISTRIBUTION WITHIN CANADA (Approval for further dissemination of the document. Security classification must also be considered.)

Public release

11b. FUTURE DISTRIBUTION OUTSIDE CANADA (Approval for further dissemination of the document. Security classification must also be considered.)

12. KEYWORDS, DESCRIPTORS or IDENTIFIERS (Use semi-colon as a delimiter.)

collaborative sensemaking; Teamwork/Collaboration; Intelligence (Direction, Collection, Processing, Dissemination); Software Design and Architecture

13. ABSTRACT/RÉSUMÉ (When available in the document, the French version of the abstract must be included here.)

ABSTRACT:
All-source analysts collaborate with each other on various information requirements. They receive different types of information from various collators, integrate and relate those information items to produce intelligence, and then share the new intelligence items with their peers.
Sharik (SHAring Resources, Information, and Knowledge) is a web-based tool aimed at supporting collaborative sensemaking among all-source intelligence analysts in distributed command and control centers. The tool has been designed, developed, and evaluated at DRDC Toronto.
The primary goal of this tool is to support analysts in producing, and more importantly sharing, new intelligence pieces with their teammates while retaining a high situational awareness of the intelligence production's status. Sharik's features support different stages of the intelligence cycle including direction, collection, processing, and dissemination.
This report describes the Design Proposal for Sharik 3.0. The report was prepared by the University of Toronto Engineering students through their University of Toronto Institute for Multidisciplinary Design & Innovation (UT-IMDI) Capstone Design Project course. The report is focused on the proposed improvements to Sharik 2.0.

RESUME:
Les analystes toutes sources collaborent entre eux afin de répondre à divers besoins en matière d'information. Ils reçoivent et intègrent divers types d'information de différents compilateurs et établissent des liens entre les éléments d'information pour en obtenir du renseignement dont ils partageront ensuite les éléments avec leurs pairs.
Sharik (SHAring Resources, Information and Knowledge, soit le partage des ressources, de l'information et des connaissances) est un outil Web qui favorise le raisonnement collaboratif chez les analystes du renseignement toutes sources disséminés dans les centres de commandement et contrôle. L'outil a été conçu, développé et évalué à RDDC Toronto.
L'objectif principal de cet outil est d'aider les analystes à produire, certes, mais surtout à partager de nouveaux éléments d'information avec leurs coéquipiers, tout en conservant une connaissance élevée de la situation quant à l'état de la production de renseignement. L'outil Sharik permet de répartir le cycle du renseignement en différentes étapes, dont l'orientation, la collecte, le traitement et la diffusion.
Ce rapport décrit la proposition de conception pour Sharik 3.0. Le rapport a été préparé par les étudiants en génie de l'Université de Toronto dans le cadre de leur cours de projet de conception Capstone de l'Institut pour la conception multidisciplinaire et l'innovation (UT-IMDI) de l'Université de Toronto. Le rapport est axé sur les améliorations proposées à Sharik 2.0.