Low Level Design

Introduction

About RUSH

The Kenya Rural Urban Sanitation and Hygiene (RUSH) platform is an advanced and comprehensive real-time monitoring and information system owned by the Ministry of Health in Kenya. This platform is designed to streamline and enhance the management of sanitation and hygiene data at both county and national levels.

One of the notable capabilities of the RUSH platform is its ability to handle large amounts of data efficiently. It supports Excel bulk upload, allowing users to upload data in bulk from Excel spreadsheets, which can significantly expedite the data entry process. Additionally, the platform features a web-form batch submission functionality, enabling users to submit multiple data entries through a user-friendly web-based interface.

To ensure data accuracy and reliability, the RUSH platform incorporates a data review and approval hierarchy between administrative levels. This means that data entered into the system undergoes a rigorous review process, where it is checked and approved by designated personnel at various administrative levels. This hierarchical approach ensures that data is thoroughly reviewed and validated before being utilised for analysis and decision-making.

Another significant aspect of the RUSH platform is its visualization capabilities. The platform follows the Joint Monitoring Program (JMP) standard and the RUSH (Rural Urban Sanitation) standard when presenting data visually. By adhering to these standards, the platform ensures consistency and comparability in data visualization across different geographical areas and time periods. The visualizations generated by the platform help in understanding trends, patterns, and gaps in sanitation and hygiene metrics, providing valuable insights for policymakers, stakeholders, and researchers.

The purpose of RUSH Platform

The purpose of the Kenya Rural Urban Sanitation and Hygiene (RUSH) platform is to support effective monitoring, management, and improvement of sanitation and hygiene practices in Kenya. It serves as a comprehensive information system owned by the Ministry of Health, aiming to address the challenges and gaps in sanitation and hygiene by providing reliable data, analysis, and visualization tools.

1. **Data Collection and Aggregation**: The RUSH platform serves as a centralised repository for collecting and aggregating both quantitative and qualitative data related to sanitation and hygiene practices. It allows for data collection at the county and national levels, ensuring comprehensive coverage and representation of diverse geographical areas.
2. **Real-Time Monitoring**: The platform operates in real-time, enabling timely monitoring of sanitation and hygiene indicators. This real-time monitoring helps identify emerging trends, gaps, and challenges, allowing for prompt intervention and decision-making.
3. **Data Analysis and Insights**: The RUSH platform facilitates data analysis, allowing policymakers and stakeholders to gain valuable insights into the state of sanitation and hygiene practices across different regions and demographics. By analising the collected data, trends, patterns, and areas of improvement can be identified, contributing to evidence-based decision-making and targeted interventions.
4. **Reporting and Visualization**: The platform enables the generation of reports and visualizations based on the collected data. The reports provide a comprehensive overview of the sanitation and hygiene situation, highlighting key indicators, challenges, and progress. The visualizations, following the JMP and RUSH standards, make complex data easily understandable, aiding in communication and knowledge dissemination.
5. **Decision Support**: The RUSH platform acts as a decision support system, providing policymakers, health officials, and other stakeholders with the necessary information to formulate policies, design interventions, and allocate resources effectively. The data-driven insights and visualizations empower decision-makers to prioritize areas for improvement, target resources where they are most needed, and track progress over time.
6. **Collaboration and Accountability**: The platform enhances collaboration between different administrative levels and stakeholders involved in sanitation and hygiene management. It establishes a data review and approval hierarchy, ensuring the accuracy and reliability of data. By promoting transparency and accountability, the platform facilitates coordinated efforts towards achieving national and international targets related to sanitation and hygiene.
7. **Continuous Improvement**: The RUSH platform can be continually updated and enhanced to align with evolving needs and priorities. As new data sources, indicators, or best practices emerge, the platform can be adapted to incorporate these changes, ensuring that it remains a relevant and effective tool for monitoring and managing sanitation and hygiene in Kenya.

By leveraging technology and real-time data, the platform aims to contribute to better health outcomes, improved living conditions, and sustainable development in both rural and urban areas of the country.

Functional Overview

The Kenya Rural Urban Sanitation and Hygiene (RUSH) platform is a comprehensive real-time monitoring and information system owned by the Ministry of Health. It serves as a centralized platform for capturing, analising, and visualizing sanitation and hygiene data at the national, county, sub-county, and ward levels. The platform provides various functionalities to facilitate data collection, analysis, reporting, and visualization, empowering decision-makers with timely and accurate information.

The RUSH platform promotes collaboration and accountability by fostering engagement between different administrative levels and stakeholders involved in sanitation and hygiene management. It acts as a decision support system, providing policymakers and health officials with the necessary information to formulate policies, design interventions, and allocate resources effectively. Additionally, the platform encourages continuous improvement by being adaptable to changing needs and priorities, accommodating new data sources, indicators, and best practices.

To ensure data accuracy and reliability, the RUSH platform incorporates a robust data review and approval hierarchy between administrative levels. This hierarchical approach guarantees that data is thoroughly reviewed, validated, and approved by designated personnel, enhancing the credibility and quality of the information within the system.

> the RUSH platform's functional overview highlights its role as a comprehensive system for data collection, analysis, reporting, and visualization.

Data Collection and Management

- The RUSH platform enables users to input data through user-friendly forms, allowing for efficient data collection.
- Users can make use of features like Excel bulk upload to upload a large amount of data in a structured manner, facilitating data entry and saving time.
- The platform supports data validation, ensuring the accuracy and integrity of the collected data.
- Data entries are associated with the respective administrative levels, allowing for easy filtering and analysis based on administrative geographical hierarchy.

Approval Hierarchy

- The RUSH platform incorporates an approval hierarchy system to ensure data accuracy and control.
- Administrators at each level have the authority to approve or reject data entries based on their jurisdiction.
- Approvers can review and make necessary edits or corrections to the data before approving or rejecting it.
- The approval hierarchy helps maintain data quality and integrity by involving multiple levels of review and verification.

User Roles and Access Control

- The RUSH platform implements a role-based access control system to manage user permissions and access levels.
- Users are assigned roles based on their responsibilities and administrative levels.
- Each role has specific page access permissions, allowing users to perform relevant tasks within their assigned administrative level.
- The platform ensures secure access and proper segregation of duties by granting appropriate permissions to users based on their roles.

Visualisations and Reports

- The platform provides visualisations following the Joint Monitoring Programme (JMP) and RUSH standards.
- Visualisations include charts, aggregates, tables, and advanced filters.
- These visualisations allow users to gain insights into the collected data, track trends, and generate reports.
- Reports can be generated based on selected criteria, such as administrative level, time period, and specific indicators.
- The platform offers export functionalities, allowing users to download reports or visualisations for further analysis or sharing.

Design Considerations

The design of the RUSH platform incorporates several key considerations to ensure its effectiveness in addressing the challenges and requirements of managing sanitation and hygiene practices in Kenya. Some of the design considerations of the RUSH platform include:

1. **Data Aggregation and Integration:** The RUSH platform is designed to aggregate both quantitative and qualitative data from various sources and administrative levels. It integrates data from county and national levels, allowing for comprehensive and unified data management. This design consideration enables a holistic view of sanitation and hygiene practices across different geographical areas.
2. **Real-time Monitoring and Reporting:** The platform emphasise real-time monitoring of sanitation and hygiene indicators. It provides timely updates on data collection, analysis, and reporting, enabling prompt interventions and decision-making. This design consideration ensures that stakeholders have access to the most up-to-date information to address emerging challenges effectively.
3. **User-Friendly Interface:** The RUSH platform features a user-friendly interface that enhances usability and accessibility. It is designed with intuitive navigation, clear visual cues, and streamlined workflows. This consideration enables users of varying technical backgrounds to easily navigate the platform and perform tasks efficiently.
4. **Role-Based Access and Permissions:** The platform employs role-based access control, assigning different levels of access and permissions based on user roles and administrative levels. This design consideration ensures data security, privacy, and appropriate data management by allowing users to access only the functionalities and

data relevant to their roles and responsibilities.

5. **Data Validation and Approval Hierarchy:** The RUSH platform incorporates a data validation process and approval hierarchy to ensure data accuracy and reliability. Appropriate users at different administrative levels review, validate, and approve the data, maintaining data integrity throughout the platform.
6. **Standardized Visualizations:** The platform follows standardized visualization practices, including the Joint Monitoring Programme (JMP) standard and the RUSH standard. This design consideration ensures consistency and comparability in data visualizations, allowing for meaningful insights and effective communication of information across different regions and time periods.
7. **Scalability and Adaptability:** The design of the RUSH platform takes into account its scalability and adaptability. It is built to accommodate a growing volume of data and changing requirements over time. This consideration ensures that the platform can evolve and meet the changing needs of sanitation and hygiene management in Kenya.
8. **Integration of Existing Systems:** The design of the RUSH platform takes into consideration the integration of existing systems and data sources. It aims to leverage and integrate with other relevant platforms, databases, and information systems to facilitate data exchange, interoperability, and collaboration.

These design considerations are aimed at creating a robust, user-friendly, and scalable platform that effectively supports data management, analysis, reporting, and decision-making for improved sanitation and hygiene practices in Kenya.

Architecture

Class Diagrams

Class Functions

User Roles

The RUSH platform offers a range of user roles, each with its own set of capabilities and responsibilities. The Super Admin holds the highest level of administrative authority at the national level and oversees the overall operation of the platform. County Admins have the responsibility of managing the platform within their respective counties, while Data Approvers review and approve data at the sub-county level. Data Entry Staff are responsible for collecting data at the ward level, ensuring that information is captured accurately at the grassroots level. Additionally, Institutional Users have access to view and download data from all counties, facilitating research and analysis.

These user roles, aligned with administrative levels, contribute to the effective management of sanitation and hygiene data. By assigning specific roles and access privileges, the RUSH platform ensures that data is collected, validated, and utilised appropriately. This promotes accountability, collaboration, and evidence-based decision-making, leading to improved sanitation and hygiene practices throughout Kenya.

In the following sections is the detailed descriptions of each user role, outlining their specific capabilities, page access, administration levels, and responsibilities. Understanding the functions and responsibilities of these user roles is vital to effectively utilising the RUSH platform and harnessing its full potential for transforming sanitation and hygiene practices in Kenya.

1. **Super Admin:** The Super Admin holds the highest level of administrative authority in the RUSH platform at the national level. They have access to all functionalities and pages, including user management, data control, visualisation, questionnaires, approvals, and reports. As the overall national administrator, their responsibilities encompass assigning roles to County Admins, managing the organisation's settings, and overseeing the platform's operations. The Super Admin plays a crucial role in ensuring the smooth functioning and effective utilisation of the RUSH platform nationwide.
2. **County Admin:** County Admins are responsible for overseeing the RUSH platform at the county level. They possess extensive access to functionalities and pages, including user management, data control, visualisation, questionnaires, approvals, and reports. Their primary role involves managing and coordinating the platform's operations within their respective counties. This includes assigning roles to Sub County RUSH Admins (Approvers) operating at the sub-county level, who play a crucial role in data management and approval. County Admins act as key facilitators in ensuring efficient and accurate data collection and analysis within their counties.
3. **Data Approver:** Data Approvers hold the responsibility of giving final approval to the data submitted from their respective sub-counties. Operating at the sub-county administrative level, they possess access to functionalities and pages such as data control, visualisation, approvals, questionnaires, and reports. Data Approvers play a critical role in reviewing and validating data submitted by Data Entry Staff from their areas of jurisdiction. They have the authority to edit or return data for correction, ensuring data accuracy and reliability within their assigned sub-counties.
4. **Data Entry Staff:** Data Entry Staff operate at the ward administrative level and are responsible for collecting data from the communities or villages assigned to them. They have access to functionalities and pages related to data entry, form submissions, data control, visualisation, and reports. Data Entry Staff play an essential role in gathering accurate and comprehensive data at the grassroots level, ensuring that the RUSH platform captures information directly from the targeted areas. Their diligent data collection efforts contribute to the overall effectiveness and reliability of the sanitation and hygiene data within the platform.
5. **Institutional User:** Institutional Users have access to functionalities and pages such as profile management, visualisation, and reports. They can view and download data from all counties within the RUSH platform. Institutional Users do not possess administrative privileges but play a vital role in accessing and utilising the data for research, analysis, and decision-making purposes. Their ability to access data from multiple administrative

levels ensures comprehensive insights and contributes to informed actions and interventions in the field of sanitation and hygiene.

Administrative Levels

The administrative levels within the RUSH platform are of utmost importance as they serve as a fundamental backbone for various components within the system. These administrative levels, provided by the Ministry of Health, play a crucial role in user management, data organisation, and the establishment of approval hierarchy rules. As such, this master list of administrative levels stands as a critical component that needs to be accurately provided by the Ministry of Health.

The administrative levels serve as a key reference for assigning roles and access privileges to users. Users are associated with specific administrative levels based on their responsibilities and jurisdiction. The administrative levels determine the data organisation structure, allowing for effective data aggregation, review, and approval processes. The approval hierarchy rules are established based on these administrative levels, ensuring proper authorisation and validation of submitted data. Additionally this allows for effective data visualisation, filtering, and analysis based on administrative boundaries.

The administrative levels consist of distinct administrative names, level names, and unique identifiers, allowing for easy identification and filtering of data points within the platform.

1. **National:** The National level represents the highest administrative level within the RUSH platform. It encompasses the entire country of Kenya and serves as the top-level jurisdiction for data management, coordination, and decision-making.
2. **County:** The County level represents the second administrative level within the RUSH platform. It corresponds to the various counties in Kenya and acts as a primary jurisdiction for data collection, management, and implementation of sanitation and hygiene initiatives.
3. **Sub-County:** The Sub-County level represents the third administrative level within the RUSH platform. It corresponds to the sub-county divisions within each county and serves as a localised jurisdiction for data collection, review, and approval processes.
4. **Ward:** The Ward level represents the fourth administrative level within the RUSH platform. It corresponds to the wards or smaller subdivisions within each sub-county. Wards act as the grassroots level of data collection, ensuring that data is collected at the most localised and community-specific level.

Here's an explanation of the models and their relationships:

1. **Levels Model:**
   - The Levels model represents the administrative levels within the RUSH platform.
   - Each instance of the Levels model corresponds to a specific administrative level, such as national, county, sub-county, or ward.
   - The model includes fields such as **name** and **level**.
   - The **name** field stores the name or label for the administrative level, as the explained administrative level above.

   - The level field stores the numerical representation of the administrative level, with lower values indicating higher levels of administration.
2. **Administration Model:**
   - The Administration model represents administrative entities within the RUSH platform.
   - Each instance of the Administration model corresponds to a specific administrative entity, such as a county or sub-county.
   - The model includes fields such as **parent, code, level, name,** and **path**.
   - The **parent** field establishes a foreign key relationship with the Administration model itself, representing the parent administrative entity.
   - The **code** field stores a unique identifier or code for the administrative entity that comes from shapefile.
   - The **level** field establishes a foreign key relationship with the Levels model, indicating the administrative level associated with the entity.
   - The **name** field stores the name or label for the administrative entity.
   - The **path** field stores the hierarchical path or location of the administrative entity within the administrative structure.

Functionality:

- The Levels model allows for the definition and categorisation of different administrative levels within the RUSH platform.
- The Administration model represents specific administrative entities, such as counties or sub-counties, and their relationships with higher-level entities.
- The parent field enables the establishment of hierarchical relationships between administrative entities, creating a structure that reflects the administrative hierarchy in the system.
- The level field associates each administrative entity with a specific administrative level, providing a standardised way to categorise and organise entities based on their level.
- The code field allows for unique identification or labeling of administrative entities, facilitating easy referencing and searchability.
- The name field stores the name or label of each administrative entity, providing a human-readable identifier for easy identification.
- The path field stores the hierarchical path or location of an administrative entity within the administrative structure, aiding in navigation and hierarchical querying.

Forms

Forms play a vital role in the RUSH platform, serving as a fundamental component for collecting data related to sanitation and hygiene practices. They are designed to capture specific information necessary for monitoring and evaluating sanitation initiatives at various administrative levels.

Importance of Forms:

1. **Data Collection:** Forms are designed to capture relevant data regarding sanitation and hygiene practices. They ensure that standardised information is collected consistently

across different administrative levels.
2. **Information Management:** Forms enable the organised storage and retrieval of data related to sanitation and hygiene practices. The collected data can be accessed, analised, and visualised for informed decision-making and policy formulation.
3. **Monitoring and Evaluation:** By collecting data through forms, the RUSH platform facilitates ongoing monitoring and evaluation of sanitation initiatives. This helps measure progress, identify challenges, and make data-driven decisions to improve sanitation and hygiene practices.
4. **Data Consistency and Standardisation:** With questionnaire definitions and question attributes, forms ensure consistency and standardisation in data collection. This promotes reliable analysis and comparison of data across different regions and time periods.
5. **Approval Workflow:** Forms incorporate approval rules and assignments, allowing designated administrators to review and approve data submitted through the platform. This ensures data quality and compliance with established standards.
6. **User Assignments:** The platform assigns specific forms to individual users, enabling targeted data collection responsibilities. This streamlines the data collection process and ensures accountability.
7. **Integration with Other Components:** Forms are integrated with other platform components such as question groups, question attributes, and options. This enhances the flexibility and customisation of data collection based on specific requirements.

Questions and Question Groups within Forms

Questions and question groups are essential components that contribute to the structured organisation and systematic data collection within forms. These components are interconnected and play a significant role in capturing information related to sanitation and hygiene practices.

1. **Forms Model**
   - The Forms represents individual forms within the RUSH platform.
   - Each form has a unique `name`, `version`, `uuid`, and `type` ("County" or "National").
   - The model establishes relationships with other models to facilitate data approval, question grouping, and user assignments.
   - Forms serve as the container for questions and question groups, defining the overall structure and context for data collection.
   - Each form is associated with specific questions and question groups that collectively capture data for a particular purpose, such as county-level or national-level sanitation assessments.
2. **Question Groups Model**
   - The Question Group represents a grouping mechanism for related questions within a form.
   - Question groups are an organisational unit within a form that groups together questions with a common theme or topic.
   - Each question group is associated with a specific form and has a unique name.
   - The order of question groups determines the sequence or presentation of these groups within the form.
3. **Questions Model**

- The Questions model represents individual questions within a form.
- Questions are associated with a specific form and question group, defining their position and relationship within the form's structure.
- Each question captures specific data points related to sanitation and hygiene practices.
- Questions can have various types (e.g., **administration (cascade), text, number, option, multiple option, geo, date**) and properties (e.g., **required, rule, dependency, and api for cascade type of question**).
- The properties of questions are defined within the context of the question group and form they belong to.

> Cascade type of question has different api call properties for each users depends on the access of the administrative of so users can only fill the form within their administrative area

Form Data

the Form Data and Answers models work together to capture, store, and associate form data and the corresponding answers within the RUSH platform.

1. **Form Data Model**
   - When a user fills out a form in the RUSH platform, the entered data is captured and stored as form data.
   - The Form Data model represents a specific data entry for a form within the platform.
   - Each instance of the Form Data model corresponds to a unique submission of a form by a user.
   - The Form Data model includes information such as the **form name, version, administration level, geographical data**, and **timestamps** for creation and updates.
   - By storing form data, the RUSH platform maintains a record of each user's submission and enables the tracking of changes and updates over time.
   - The form data is associated with the relevant form through a foreign key relationship, allowing easy retrieval and analysis of the submitted information.
2. **Answers Model**
   - Within each form data entry, the user provides answers to the questions included in the form.
   - The Answers model represents individual answers for specific questions within a form data entry.
   - Each answer in the Answers model is associated with a particular question and the corresponding form data entry.
   - The model includes fields such as the **answer value, name, options** (if applicable), and **timestamps** for creation and updates.
   - By storing answers as separate instances, the RUSH platform retains the granularity of data, allowing for detailed analysis of each answer within the form data.
   - The answers are linked to the form data and questions through foreign key relationships, facilitating easy retrieval and analysis of specific answers within a

given form data entry.

Functionality:

- When a user submits a form, the entered data is processed and saved as a new instance of the Form Data model, representing a unique data entry for that form.
- The associated answers for each question in the form are stored as instances of the Answers model, linked to the corresponding form data entry and question.
- The form data and answers are stored in the database, providing a comprehensive record of the submitted information.
- This stored data can be accessed, retrieved, and analised for various purposes, such as monitoring and evaluating sanitation and hygiene practices, generating reports, and informing decision-making processes.
- The relationship between form data and answers allows for flexible querying and analysis, enabling the platform to generate insights and visualise trends based on the collected data.

Class Overview

Class Name	Class Notes
Organisation	Organisation(id, name)
OrganisationAttribute	OrganisationAttribute(id, organisation, type)
SystemUser	SystemUser(id, password, last_login, is_superuser, email, date_joined, first_name, last_name, phone_number, designation, trained, updated, deleted_at, organisation)
Levels	Levels(id, name, level)
Administration	Administration(id, parent, code, level, name, path)
Access	Access(id, user, administration, role)
Forms	Forms(id, name, version, uuid, type)
FormApprovalRule	FormApprovalRule(id, form, administration)
FormApprovalAssignment	FormApprovalAssignment(id, form, administration, user, updated)
QuestionGroup	QuestionGroup(id, form, name, order)
Questions	Questions(id, form, question_group, order, text, name, type, meta, required, rule, dependency, api, extra)

QuestionOptions	QuestionOptions(id, question, order, code, name, other)
UserForms	UserForms(id, user, form)
QuestionAttribute	QuestionAttribute(id, name, question, attribute, options)
ViewJMPCriteria	ViewJMPCriteria(id, form, name, criteria, level, score)
FormData	FormData(id, name, form, administration, geo, created_by, updated_by, created, updated)
PendingDataBatch	PendingDataBatch(id, form, administration, user, name, uuid, file, approved, created, updated)
PendingDataBatchComments	PendingDataBatchComments(id, batch, user, comment, created)
PendingFormData	PendingFormData(id, name, form, data, administration, geo, batch, created_by, updated_by, created, updated)
PendingDataApproval	PendingDataApproval(id, batch, user, level, status)
PendingAnswers	PendingAnswers(id, pending_data, question, name, value, options, created_by, created, updated)
PendingAnswerHistory	PendingAnswerHistory(id, pending_data, question, name, value, options, created_by, created, updated)
Answers	Answers(id, data, question, name, value, options, created_by, created, updated)
AnswerHistory	AnswerHistory(id, data, question, name, value, options, created_by, created, updated)
ViewPendingDataApproval	ViewPendingDataApproval(id, status, user, level, batch, pending_level)
ViewDataOptions	ViewDataOptions(id, data, administration, form, options)
ViewOptions	ViewOptions(id, data, administration, question, answer, form, options)
ViewJMPData	ViewJMPData(id, data, path, form, name, level, matches, score)
ViewJMPCount	ViewJMPCount(id, path, form, name, level, total)
Jobs	Jobs(id, task_id, type, status, attempt, result, info, user, created, available)
DataCategory	DataCategory(id, name, data, form, options)

Task	Task(id, name, func, hook, args, kwargs, result, group, started, stopped, success, attempt_count)
Success	Success(id, name, func, hook, args, kwargs, result, group, started, stopped, success, attempt_count)
Failure	Failure(id, name, func, hook, args, kwargs, result, group, started, stopped, success, attempt_count)
Schedule	Schedule(id, name, func, hook, args, kwargs, schedule_type, minutes, repeats, next_run, cron, task, cluster)
OrmQ	OrmQ(id, key, payload, lock)

Database Overview

Main Tables

access

pos	table	column	null	dtype	len	default
1	access	id	NO	bigint		access_id_seq
2	access	role	NO	int		
3	access	administration_id	NO	bigint		
4	access	user_id	NO	bigint		

administrator

pos	table	column	null	dtype	len	default
1	administrator	id	NO	bigint		administrator_id_seq
2	administrator	code	YES	character varying	255	
3	administrator	name	NO	text		
4	administrator	level_id	NO	bigint		
5	administrator	parent_id	YES	bigint		
6	administrator	path	YES	text		

answer

pos	table	column	null	dtype	len	default

1	answer	id	NO	bigint		answer_id_seq
2	answer	name	YES	text		
3	answer	value	YES	double		
4	answer	options	YES	jsonb		
5	answer	created	NO	tz timestamp		
6	answer	updated	YES	tz timestamp		
7	answer	created_by_id	NO	bigint		
8	answer	data_id	NO	bigint		
9	answer	question_id	NO	bigint		

answer_history

pos	table	column	null	dtype	len	default
1	answer_history	id	NO	bigint		answer_history_id_seq
2	answer_history	name	YES	text		
3	answer_history	value	YES	double		
4	answer_history	options	YES	jsonb		
5	answer_history	created	NO	tz timestamp		
6	answer_history	updated	YES	tz timestamp		
7	answer_history	created_by_id	NO	bigint		
8	answer_history	data_id	NO	bigint		
9	answer_history	question_id	NO	bigint		

batch

pos	table	column	null	dtype	len	default
1	batch	id	NO	bigint		batch_id_seq
2	batch	name	NO	text		
3	batch	uuid	YES	uuid		
4	batch	file	YES	character varying	200	
5	batch	created	NO	tz timestamp		
6	batch	updated	YES	tz timestamp		
7	batch	administration_id	NO	bigint		
8	batch	form_id	NO	bigint		

pos	table	column	null	dtype	len	default
9	batch	user_id	NO	bigint		
10	batch	approved	NO	bool		

batch_comment

pos	table	column	null	dtype	len	default
1	batch_comment	id	NO	bigint		batch_comment_id_seq
2	batch_comment	comment	NO	text		
3	batch_comment	created	NO	tz timestamp		
4	batch_comment	batch_id	NO	bigint		
5	batch_comment	user_id	NO	bigint		

data

pos	table	column	null	dtype	len	default
1	data	id	NO	bigint		data_id_seq
2	data	name	NO	text		
3	data	geo	YES	jsonb		
4	data	created	NO	tz timestamp		
5	data	updated	YES	tz timestamp		
6	data	administration_id	NO	bigint		
7	data	created_by_id	NO	bigint		
8	data	form_id	NO	bigint		
9	data	updated_by_id	YES	bigint		

form

pos	table	column	null	dtype	len	default
1	form	id	NO	bigint		form_id_seq
2	form	name	NO	text		
3	form	version	NO	int		
4	form	uuid	NO	uuid		
5	form	type	YES	int		

form_approval_assignment

pos	table	column	null	dtype	len	default
1	form_approval_assignment	id	NO	bigint		form_approval_assignment_id_seq
2	form_approval_assignment	updated	YES	tz timestamp		
3	form_approval_assignment	administration_id	NO	bigint		
4	form_approval_assignment	form_id	NO	bigint		
5	form_approval_assignment	user_id	NO	bigint		

form_approval_rule

pos	table	column	null	dtype	len	default
1	form_approval_rule	id	NO	bigint		form_approval_rule_id_seq
2	form_approval_rule	administration_id	NO	bigint		
3	form_approval_rule	form_id	NO	bigint		

jobs

pos	table	column	null	dtype	len	default
1	jobs	id	NO	bigint		jobs_id_seq
2	jobs	type	NO	int		
3	jobs	status	NO	int		
4	jobs	attempt	NO	int		
5	jobs	result	YES	text		
6	jobs	info	YES	jsonb		
7	jobs	created	NO	tz timestamp		
8	jobs	available	YES	tz timestamp		
9	jobs	user_id	NO	bigint		
10	jobs	task_id	YES	character varying	50	

levels

pos	table	column	null	dtype	len	default
1	levels	id	NO	bigint		levels_id_seq

pos	table	column	null	dtype	len	default
2	levels	name	NO	character varying	50	
3	levels	level	NO	int		

option

pos	table	column	null	dtype	len	default
1	option	id	NO	bigint		option_id_seq
2	option	order	YES	bigint		
3	option	code	YES	character varying	255	
4	option	name	NO	text		
5	option	other	NO	bool		
6	option	question_id	NO	bigint		

organisation

pos	table	column	null	dtype	len	default
1	organisation	id	NO	bigint		organisation_id_seq
2	organisation	name	NO	character varying	255	

organisation_attribute

pos	table	column	null	dtype	len	default
1	organisation_attribute	id	NO	bigint		organisation_attribute_id_seq
2	organisation_attribute	type	NO	int		
3	organisation_attribute	organisation_id	NO	bigint		

pending_answer

pos	table	column	null	dtype	len	default
1	pending_answer	id	NO	bigint		pending_answer_id_seq
2	pending_answer	name	YES	text		
3	pending_answer	value	YES	double		
4	pending_answer	options	YES	jsonb		

pos	table	column	null	dtype	len	default
5	pending_answer	created	NO	tz timestamp		
6	pending_answer	updated	YES	tz timestamp		
7	pending_answer	created_by_id	NO	bigint		
8	pending_answer	pending_data_id	NO	bigint		
9	pending_answer	question_id	NO	bigint		

pending_answer_history

pos	table	column	null	dtype	len	default
1	pending_answer_history	id	NO	bigint		pending_answer_history_id_seq
2	pending_answer_history	name	YES	text		
3	pending_answer_history	value	YES	double		
4	pending_answer_history	options	YES	jsonb		
5	pending_answer_history	created	NO	tz timestamp		
6	pending_answer_history	updated	YES	tz timestamp		
7	pending_answer_history	created_by_id	NO	bigint		
8	pending_answer_history	pending_data_id	NO	bigint		
9	pending_answer_history	question_id	NO	bigint		

pending_data

pos	table	column	null	dtype	len	default
1	pending_data	id	NO	bigint		pending_data_id_seq
2	pending_data	name	NO	text		
3	pending_data	geo	YES	jsonb		
5	pending_data	created	NO	tz timestamp		
6	pending_data	administration_id	NO	bigint		
7	pending_data	created_by_id	NO	bigint		

pos	table	column	null	dtype	len	default
8	pending_data	data_id	YES	bigint		
9	pending_data	form_id	NO	bigint		
11	pending_data	batch_id	YES	bigint		
12	pending_data	updated	YES	tz timestamp		
13	pending_data	updated_by_id	YES	bigint		

pending_data_approval

pos	table	column	null	dtype	len	default
1	pending_data_approval	id	NO	bigint		pending_data_approval_id_seq
2	pending_data_approval	status	NO	int		
4	pending_data_approval	user_id	NO	bigint		
5	pending_data_approval	level_id	NO	bigint		
6	pending_data_approval	batch_id	NO	bigint		

question

pos	table	column	null	dtype	len	default
1	question	id	NO	bigint		question_id_seq
2	question	order	YES	bigint		
3	question	text	NO	text		
4	question	name	NO	character varying	255	
5	question	type	NO	int		
6	question	meta	NO	bool		
7	question	required	NO	bool		
8	question	rule	YES	jsonb		
9	question	dependency	YES	jsonb		
10	question	form_id	NO	bigint		
11	question	question_group_id	NO	bigint		
12	question	api	YES	jsonb		
13	question	extra	YES	jsonb		

question_attribute

pos	table	column	null	dtype	len	default
1	question_attribute	id	NO	bigint		question_attribute_id_seq
2	question_attribute	name	YES	text		
3	question_attribute	attribute	NO	int		
4	question_attribute	options	YES	jsonb		
5	question_attribute	question_id	NO	bigint		

question_group

pos	table	column	null	dtype	len	default
1	question_group	id	NO	bigint		question_group_id_seq
2	question_group	name	NO	text		
3	question_group	form_id	NO	bigint		
4	question_group	order	YES	bigint		

system_user

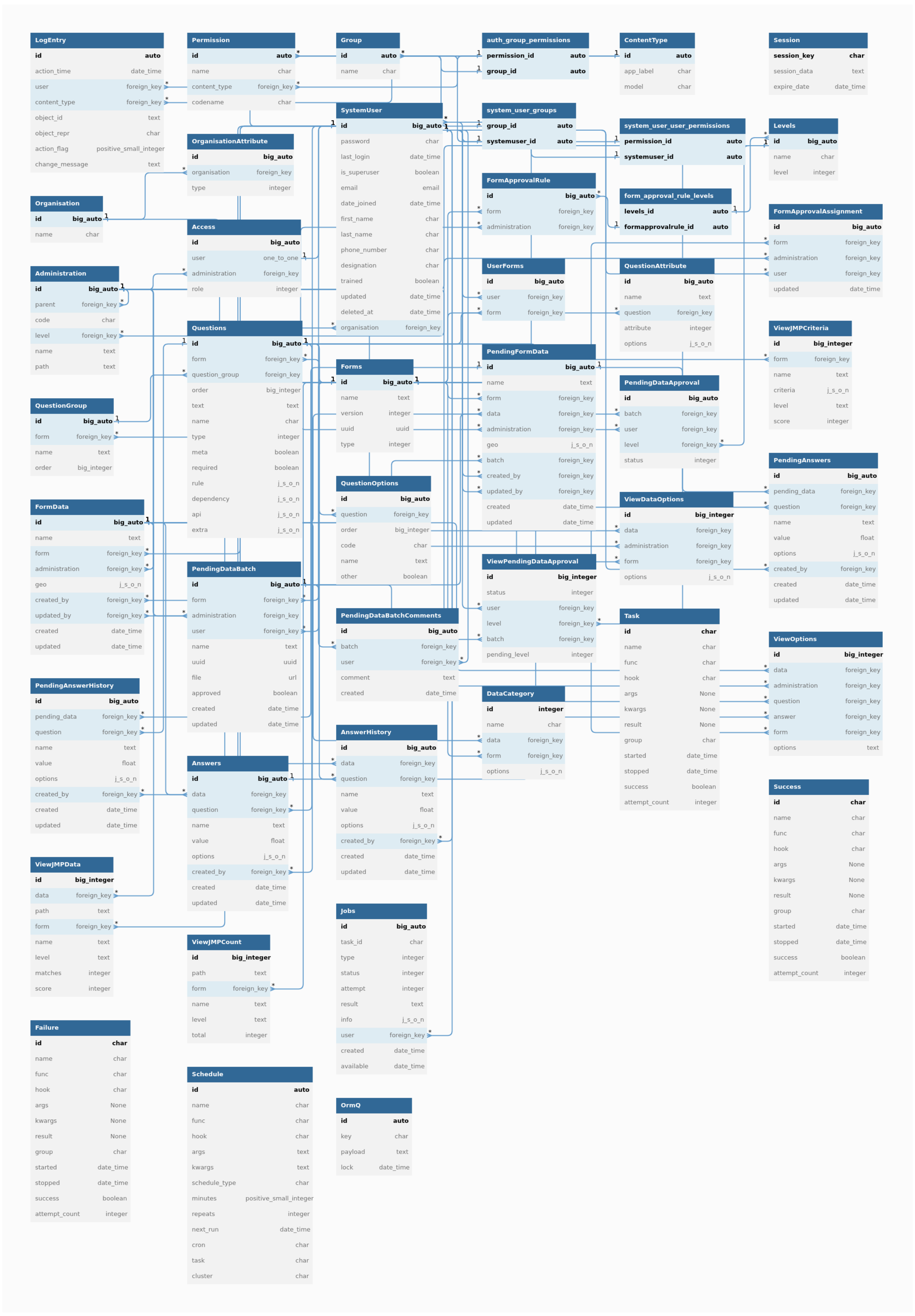
pos	table	column	null	dtype	len	default
1	system_user	id	NO	bigint		system_user_id_seq
2	system_user	password	NO	character varying	128	
3	system_user	last_login	YES	tz timestamp		
4	system_user	is_superuser	NO	bool		
5	system_user	email	NO	character varying	254	
6	system_user	date_joined	NO	tz timestamp		
7	system_user	first_name	NO	character varying	50	
8	system_user	last_name	NO	character varying	50	
9	system_user	designation	YES	character varying	50	
10	system_user	phone_number	YES	character varying	15	
11	system_user	updated	YES	tz timestamp		
12	system_user	deleted_at	YES	tz timestamp		
13	system_user	organisation_id	YES	bigint		
14	system_user	trained	NO	bool		

user_form

pos	table	column	null	dtype	len	default
1	user_form	id	NO	bigint		user_form_id_seq
2	user_form	form_id	NO	bigint		
3	user_form	user_id	NO	bigint		

Materialized Views

Relationship Diagrams



LogEntry	
id	auto
action_time	date_time
user	foreign_key
content_type	foreign_key
object_id	text
object_repr	char
action_flag	positive_small_integer
change_message	text

Permission	
id	auto
name	char
content_type	foreign_key
codename	char

Group	
id	auto
name	char

auth_group_permissions	
permission_id	auto
group_id	auto

ContentType	
id	auto
app_label	char
model	char

Session	
session_key	char
session_data	text
expire_date	date_time

SystemUser	
id	big_auto
password	char
last_login	date_time
is_superuser	boolean
email	email
date_joined	date_time
first_name	char
last_name	char
phone_number	char
designation	char
trained	boolean
updated	date_time
deleted_at	date_time
organisation	foreign_key

system_user_groups	
group_id	auto
systemuser_id	auto

system_user_user_permissions	
permission_id	auto
systemuser_id	auto

Levels	
id	big_auto
name	char
level	integer

OrganisationAttribute	
id	big_auto
organisation	foreign_key
type	integer

Organisation	
id	big_auto
name	char

FormApprovalRule	
id	big_auto
form	foreign_key
administration	foreign_key

form_approval_rule_levels	
levels_id	auto
formapprovalrule_id	auto

FormApprovalAssignment	
id	big_auto
form	foreign_key
administration	foreign_key
user	foreign_key
updated	date_time

Access	
id	big_auto
user	one_to_one
administration	foreign_key
role	integer

Administration	
id	big_auto
parent	foreign_key
code	char
level	foreign_key
name	text
path	text

UserForms	
id	big_auto
user	foreign_key
form	foreign_key

QuestionAttribute	
id	big_auto
name	text
question	foreign_key
attribute	integer
options	j_s_o_n

ViewJMPCriteria	
id	big_integer
form	foreign_key
name	text
criteria	j_s_o_n
level	text
score	integer

Questions	
id	big_auto
form	foreign_key
question_group	foreign_key
order	big_integer
text	text
name	char
type	integer
meta	boolean
required	boolean
rule	j_s_o_n
dependency	j_s_o_n
api	j_s_o_n
extra	j_s_o_n

Forms	
id	big_auto
name	text
version	integer
uuid	uuid
type	integer

PendingFormData	
id	big_auto
name	text
form	foreign_key
data	foreign_key
administration	foreign_key
geo	j_s_o_n
batch	foreign_key
created_by	foreign_key
updated_by	foreign_key
created	date_time
updated	date_time

PendingDataApproval	
id	big_auto
batch	foreign_key
user	foreign_key
level	foreign_key
status	integer

QuestionGroup	
id	big_auto
form	foreign_key
name	text
order	big_integer

PendingAnswers	
id	big_auto
pending_data	foreign_key
question	foreign_key
name	text
value	float
options	j_s_o_n
created_by	foreign_key
created	date_time
updated	date_time

QuestionOptions	
id	big_auto
question	foreign_key
order	big_integer
code	char
name	text
other	boolean

ViewDataOptions	
id	big_integer
data	foreign_key
administration	foreign_key
form	foreign_key
options	j_s_o_n

FormData	
id	big_auto
name	text
form	foreign_key
administration	foreign_key
geo	j_s_o_n
created_by	foreign_key
updated_by	foreign_key
created	date_time
updated	date_time

PendingDataBatch	
id	big_auto
form	foreign_key
administration	foreign_key
user	foreign_key
name	text
uuid	uuid
file	url
approved	boolean
created	date_time
updated	date_time

ViewPendingDataApproval	
id	big_integer
status	integer
user	foreign_key
level	foreign_key
batch	foreign_key
pending_level	integer

PendingDataBatchComments	
id	big_auto
batch	foreign_key
user	foreign_key
comment	text
created	date_time

Task	
id	char
name	char
func	char
hook	char
args	None
kwargs	None
result	None
group	char
started	date_time
stopped	date_time
success	boolean
attempt_count	integer

ViewOptions	
id	big_integer
data	foreign_key
administration	foreign_key
question	foreign_key
answer	foreign_key
form	foreign_key
options	text

PendingAnswerHistory	
id	big_auto
pending_data	foreign_key
question	foreign_key
name	text
value	float
options	j_s_o_n
created_by	foreign_key
created	date_time
updated	date_time

DataCategory	
id	integer
name	char
data	foreign_key
form	foreign_key
options	j_s_o_n

AnswerHistory	
id	big_auto
data	foreign_key
question	foreign_key
name	text
value	float
options	j_s_o_n
created_by	foreign_key
created	date_time
updated	date_time

Answers	
id	big_auto
data	foreign_key
question	foreign_key
name	text
value	float
options	j_s_o_n
created_by	foreign_key
created	date_time
updated	date_time

Success	
id	char
name	char
func	char
hook	char
args	None
kwargs	None
result	None
group	char
started	date_time
stopped	date_time
success	boolean
attempt_count	integer

ViewJMPData	
id	big_integer
data	foreign_key
path	text
form	foreign_key
name	text
level	text
matches	integer
score	integer

ViewJMPCount	
id	big_integer
path	text
form	foreign_key
name	text
level	text
total	integer

Jobs	
id	big_auto
task_id	char
type	integer
status	integer
attempt	integer
result	text
info	j_s_o_n
user	foreign_key
created	date_time
available	date_time

Failure	
id	char
name	char
func	char
hook	char
args	None
kwargs	None
result	None
group	char
started	date_time
stopped	date_time
success	boolean
attempt_count	integer

Schedule	
id	auto
name	char
func	char
hook	char
args	text
kwargs	text
schedule_type	char
minutes	positive_small_integer
repeats	integer
next_run	date_time
cron	char
task	char
cluster	char

OrmQ	
id	auto
key	char
payload	text
lock	date_time

To generate the relationship diagram for the RUSH platform, the dbdocs.io tool is utilized. The process involves using the **django-dbml** library to generate a dbml (database markup language) file that represents the database schema and entity relationships based on the Django models.

This dbml file is then pushed to a designated location, accessible during the CI/CD pipeline. The dbdocs.io command-line tool is utilized to build the documentation using the dbml file. The process typically includes specifying the location of the dbml file and providing a project name, which may be customized based on the CI/CD environment or branch. Once the documentation is built, the resulting relationship diagram can be accessed via the generated dbdocs.io link, which provides a visual representation of the database schema and the relationships between entities within the RUSH platform.

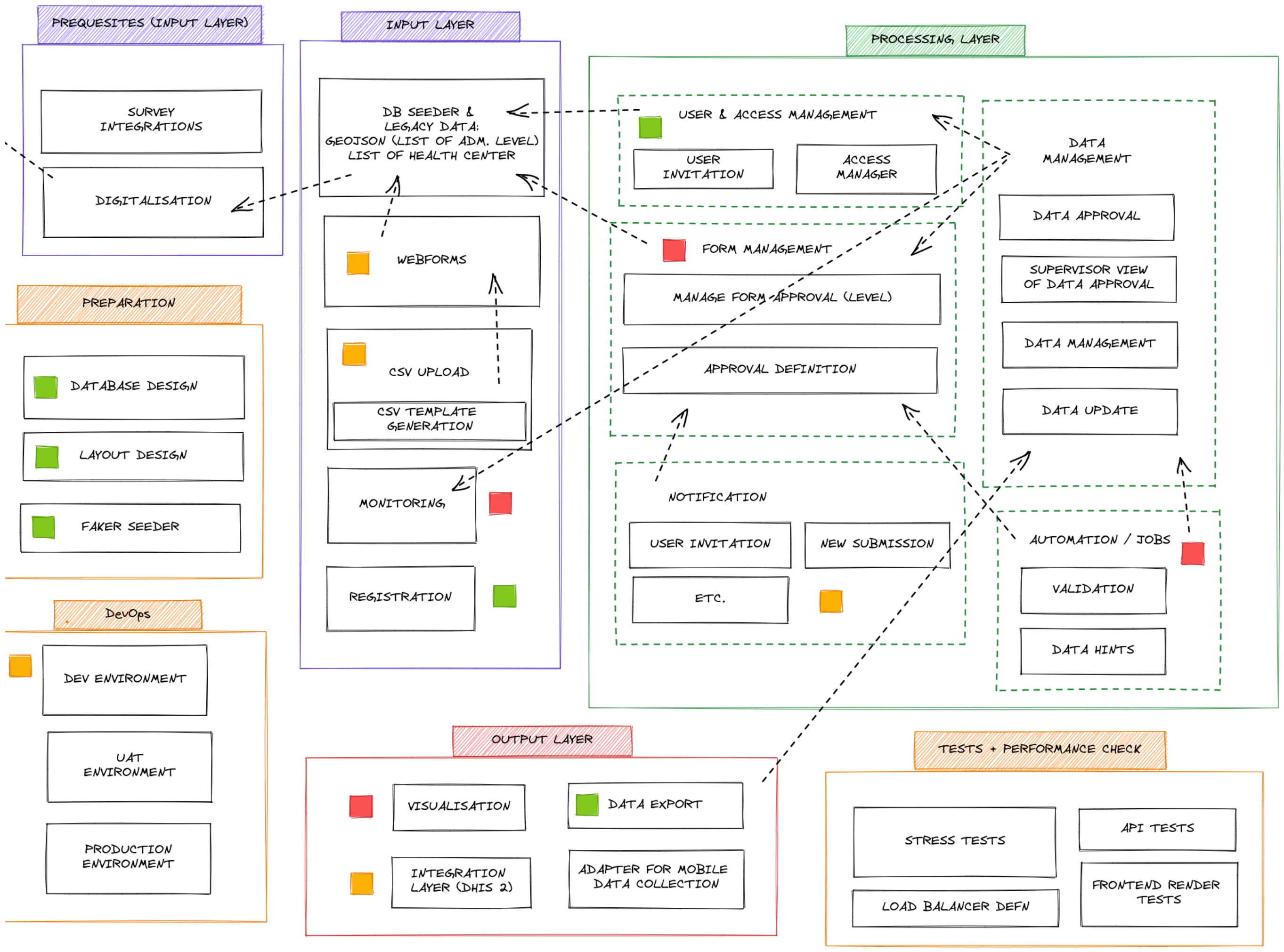
```
# Generate DBML
# https://github.com/akvo/rtmis/blob/main/backend/run-qc.sh#L22
python manage.py dbml > db.dbml

# Push DBDocs
# https://github.com/akvo/rtmis/blob/main/ci/build.sh#L116-L122
update_dbdocs() {
    if [[ "${CI_BRANCH}" ==  "main" || "${CI_BRANCH}" ==  "develop" ]]; then
        npm install -g dbdocs
        # dbdocs build doc/dbml/schema.dbml --project rtmis
        dbdocs build backend/db.dbml --project "rtmis-$CI_BRANCH"
    fi
}
```

> To view the comprehensive relationship diagram for the RUSH platform, please refer to the following link: **RUSH Platform Relationship Diagram.**

Sequence Diagrams

Data Flow Diagrams



User Interface Design

The RUSH platform incorporates a range of user interfaces designed to enhance usability, streamline workflows, and enable efficient data management and analysis. These interfaces serve as the gateway for users to interact with the platform's various features and functionalities. From the login page that grants access to authenticated users, to the dashboard providing an informative overview of key data and notifications, each interface has a specific purpose and contributes to the seamless operation of the platform.

- **Login Page:** The login page allows users to authenticate themselves and access the platform using their credentials.
- **Dashboard:** The dashboard serves as the main interface after login, providing an overview of key information, notifications, and access to different modules and functionalities.
- **Data Entry Forms:** User-friendly forms are designed for data collection, enabling users to input sanitation and hygiene data accurately and efficiently.
- **Form Management Interface:** Administrators can create, edit, and manage forms, including defining question groups, adding questions, setting validation rules, and

configuring approval workflows.
- **Data Review and Approval Interface:** This interface allows authorised users to review, edit, approve, or reject data entries based on their administrative levels and approval roles.
- **Visualisations Interface:** Each form will have a dedicated visualisation page where users can view interactive charts, graphs, tables, and maps representing the collected data for that specific form.
- **User Management Interface:** Administrators can manage user accounts, roles, and access permissions within the RUSH platform.
- **Approval Hierarchy Interface:** This interface provides a visual representation of the approval hierarchy, showcasing the different levels and roles involved in the data approval process.
- **Data Import/Export Interface:** This interface facilitates the import and export Excel files, which can be filtered by geographical administrative area and advanced filters (filter by specific input of the submission).
- **Settings and Configuration Interface:** Administrators can access and modify platform settings, including email notifications, system preferences, and integration configurations.
- **Notifications and Alerts Interface:** Users can receive important notifications, alerts, and reminders through the platform, ensuring timely communication and action.
- **User Profile Interface:** Users can view their personal information, including profile details and list of assigned forms.
- **Help and Support Interface:** This interface provides users with access to documentation, FAQs, tutorials, and support resources to assist them in using the platform effectively.
- **Data Search and Filtering Interface:** Users can search and filter data based on specific criteria, allowing them to retrieve relevant information quickly.
- **Error and Exception Handling Interface:** When errors occur, an interface can display informative error messages and provide guidance on how to resolve or report the issue.

These user interfaces collectively offer a comprehensive and intuitive user experience, facilitating efficient data entry, analysis, visualization, approval workflows, and decision-making within the RUSH platform.

> For a detailed visual representation of the user interfaces within the RUSH platform, please refer to the design interface available at the following link: **RUSH Platform Design Interface.**

This interface showcases the overall layout, design elements, and interactions that users can expect when navigating through the platform. It provides a valuable reference for understanding the visual aesthetics, information architecture, and user flow incorporated into the RUSH platform's user interfaces. By exploring the design interface, stakeholders can gain a clearer understanding of the platform's look and feel, facilitating better collaboration and alignment throughout the development process.

Error Handling

Error Handling Rules

The platform incorporates robust error handling strategies to address various types of errors that may occur during operation. Here are the key considerations for error handling in the RUSH platform:

1. **Error Logging and Monitoring:** The platform logs errors and exceptions that occur during runtime. These logs capture relevant details such as the error type, timestamp, user context, and relevant system information. Error logs enable developers and administrators to identify and troubleshoot issues efficiently, helping to improve system reliability and performance.
2. **User-Friendly Error Messages:** When errors occur, the platform provides user-friendly error messages that communicate the issue clearly and concisely. Clear error messages help users understand the problem and take appropriate actions or seek assistance. The messages may include relevant details about the error, potential solutions, and contact information for support if necessary.
3. **Graceful Degradation and Recovery:** The platform is designed to handle errors gracefully, minimising disruptions and providing fallback mechanisms where possible. For example, if a specific functionality or service becomes temporarily unavailable, the platform can display a fallback message or provide alternative options to ensure users can continue their work or access relevant information.
4. **Error Validation and Input Sanitisation:** The platform applies comprehensive input validation and sanitisation techniques to prevent and handle errors caused by invalid or malicious user input. This includes validating user-submitted data, sanitising inputs to prevent code injection or script attacks, and ensuring that data conforms to expected formats and ranges. Proper input validation reduces the risk of errors and security vulnerabilities.
5. **Exception Handling and Error Recovery:** The platform utilises exception handling mechanisms to catch and handle errors gracefully. Exceptions are caught, logged, and processed to prevent system crashes or unexpected behavior. The platform incorporates appropriate error recovery strategies, such as rolling back transactions or reverting to previous states, to maintain data integrity and prevent data loss or corruption.
6. **Error Reporting and Support Channels:** The platform provides channels for users to report errors and seek support. These channels can include contact forms, dedicated support email addresses, or a help-desk system. By offering reliable channels for error reporting and support, users can report issues promptly and receive assistance in resolving them effectively.
7. **Continuous Improvement:** The platform regularly assesses error patterns and user feedback to identify recurring issues and areas for improvement. By analising error trends, the development team can prioritise bug fixes, optimise system components, and

enhance the overall stability and reliability of the platform.

List Errors

The following section provides an overview of potential errors that may occur within the RUSH platform. While data validation plays a significant role in minimizing errors during data entry and form submissions, certain issues can still arise in other aspects of the platform's functionality. These errors encompass various areas, including authentication, authorization, file uploads, data synchronization, network connectivity, server timeouts, data import/export, data corruption, missing data, report generation, visualization, server overload, email notifications, and third-party integrations. By being aware of these potential errors, the development team can proactively address and implement proper error handling mechanisms to ensure smooth operations, enhance user experience, and maintain data integrity throughout the platform.

1. **Database Connection Error:** Failure to establish a connection with the database server, resulting in the inability to retrieve or store data.
2. **Authentication Error:** Users may encounter authentication errors when attempting to log in, indicating invalid credentials or authentication failures.
3. **Authorisation Error:** Users may encounter authorisation errors when accessing certain features or performing actions for which they do not have sufficient privileges.
4. **File Upload Error:** When uploading files, errors may occur due to file format compatibility, size limitations, or network connectivity issues.
5. **Data Synchronisation Error:** In a multi-user environment, conflicts may arise when multiple users attempt to update the same data simultaneously, leading to synchronisation errors.
6. **Network Connectivity Error:** Users may experience network connectivity issues, preventing them from accessing the platform or transmitting data.
7. **Server Timeout Error:** When processing resource-intensive tasks, such as generating complex reports or visualizations, server timeouts may occur if the process exceeds the maximum allowed execution time.
8. **Data Import/Export Error:** Errors may occur during the import or export of data, resulting in data loss, formatting issues, or mismatches between source and destination formats.
9. **Data Corruption Error:** In rare cases, data corruption may occur, leading to inconsistencies or incorrect values in the database.
10. **Missing Data Error:** Users may encounter missing data issues when attempting to retrieve or access specific records or fields that have not been properly captured or stored.
11. **Report Generation Error:** Errors may occur during the generation of reports, resulting in incomplete or inaccurate data representation or formatting issues.
12. **Visualization Error:** Issues with data visualization components, such as charts or graphs, may lead to incorrect data representation or inconsistencies in visual outputs.
13. **Server Overload Error:** During periods of high user activity or resource-intensive tasks, the server may become overloaded, causing slowdowns or system instability.

14. **Email Notification Error:** Failure to send email notifications, such as approval requests or password reset emails, may occur due to issues with the email service or configuration.
15. **Third-Party Integration Error:** Errors may arise when integrating with external services or APIs, resulting in data transfer issues or functionality limitations.

These errors represent potential issues that may arise in the RUSH platform, excluding errors already addressed by data validation measures. It's crucial to implement proper error handling and logging mechanisms to promptly identify, track, and resolve these errors, ensuring the smooth functioning of the platform.

Security Considerations

The RUSH platform incorporates multiple security measures to safeguard data, protect user privacy, and ensure secure operations across its Docker containers and cloud-based infrastructure. Here are the key security considerations in the platform:

1. **Container Security (Docker):** The Docker containers, including the Back-end and Worker containers, are designed with security in mind. The containers are configured to follow best practices such as using official base images, regularly updating dependencies, and employing secure container runtime configurations. These measures reduce the risk of vulnerabilities and unauthorised access within the containerised environment.
2. **Access Control and Authentication:** The platform implements robust access control mechanisms to ensure that only authorised users can access the system and its functionalities. User authentication, such as through the use of JWT (JSON Web Token), is employed to verify user identities and grant appropriate access based on roles and permissions. This helps prevent unauthorised access to sensitive data and functionalities.
3. **Network Security (NGINX):** The Front-end container, powered by NGINX, helps enforce security measures at the network level. NGINX can be configured to handle SSL/TLS encryption, protecting data in transit between users and the platform. It can also serve as a reverse proxy, effectively managing incoming traffic and providing an additional layer of security to prevent potential attacks.
4. **Secure Database Storage (Cloud-SQL):** The RUSH platform utilises Cloud-SQL for secure database storage. Cloud-SQL offers built-in security features, including encryption at rest and transit, role-based access control, and regular security updates. These measures help protect the integrity and confidentiality of the platform's data stored in the Cloud-SQL database.
5. **Secure File Storage (Cloud Storage Bucket):** The platform leverages Cloud Storage Bucket for secure file storage. Cloud Storage provides robust access controls, including fine-grained permissions, encryption, and auditing capabilities. This ensures that data files, such as uploaded documents, are securely stored and protected from unauthorised access. **The endpoints of file should only served by the back-end** so it also applies authentication.

6. **Security Monitoring and Auditing:** The platform implements security monitoring and auditing tools to detect and respond to potential incidents. System logs and activity records are regularly reviewed to identify any suspicious activities or breaches. Additionally, periodic security audits are conducted to assess and address potential vulnerabilities proactively.
7. **User Education and Awareness:** The platform emphasise user education and awareness regarding security best practices. Users are encouraged to follow strong password policies: **Lowercase, Numbers, Special Character, Uppercase Character, No White Space, and Minimum 8 Characters**. By promoting user security awareness, the platform strengthens overall security posture.

Performance Considerations

The RUSH platform has several performance considerations, particularly in relation to visualisation, excel data download, data upload, and validation. While these functionalities are crucial for effective data management and analysis, they can pose potential performance challenges due to the volume and complexity of the data involved. The platform takes these considerations into account to optimise performance and ensure a smooth user experience. Here are the key performance considerations:

1. **Visualisation:** Visualisations are powerful tools for data analysis and communication. However, generating complex visualisations from large datasets can be computationally intensive and may lead to performance issues. The RUSH platform employs optimisation techniques, such as efficient data retrieval, caching, and rendering algorithms, to enhance the speed and responsiveness of visualisations. It strives to strike a balance between visual richness and performance to provide users with meaningful insights without sacrificing usability.
2. **Excel Data Download:** The ability to download data in Excel format is essential for users to perform in-depth analysis and reporting. However, large datasets or complex queries can result in long download times and increased server load. To mitigate this, the RUSH platform optimises the data retrieval and export process, employing techniques such as data compression and efficient file generation. It aims to minimise download times and ensure a seamless user experience when exporting data to Excel.
3. **Data Upload and Validation:** Data upload and validation involve processing and verifying large volumes of data. This process can be time-consuming, particularly when dealing with extensive datasets or complex validation rules. The RUSH platform optimises data upload and validation processes through efficient algorithms and parallel processing techniques. It strives to expedite the data entry process while maintaining data integrity and accuracy.

To ensure optimal performance, the RUSH platform continuously monitors system performance, identifies bottlenecks, and implements performance optimisations as needed. This may involve infrastructure scaling, database optimisations, query optimisations, and caching strategies. Regular maintenance and updates are conducted to keep the platform running smoothly and efficiently.

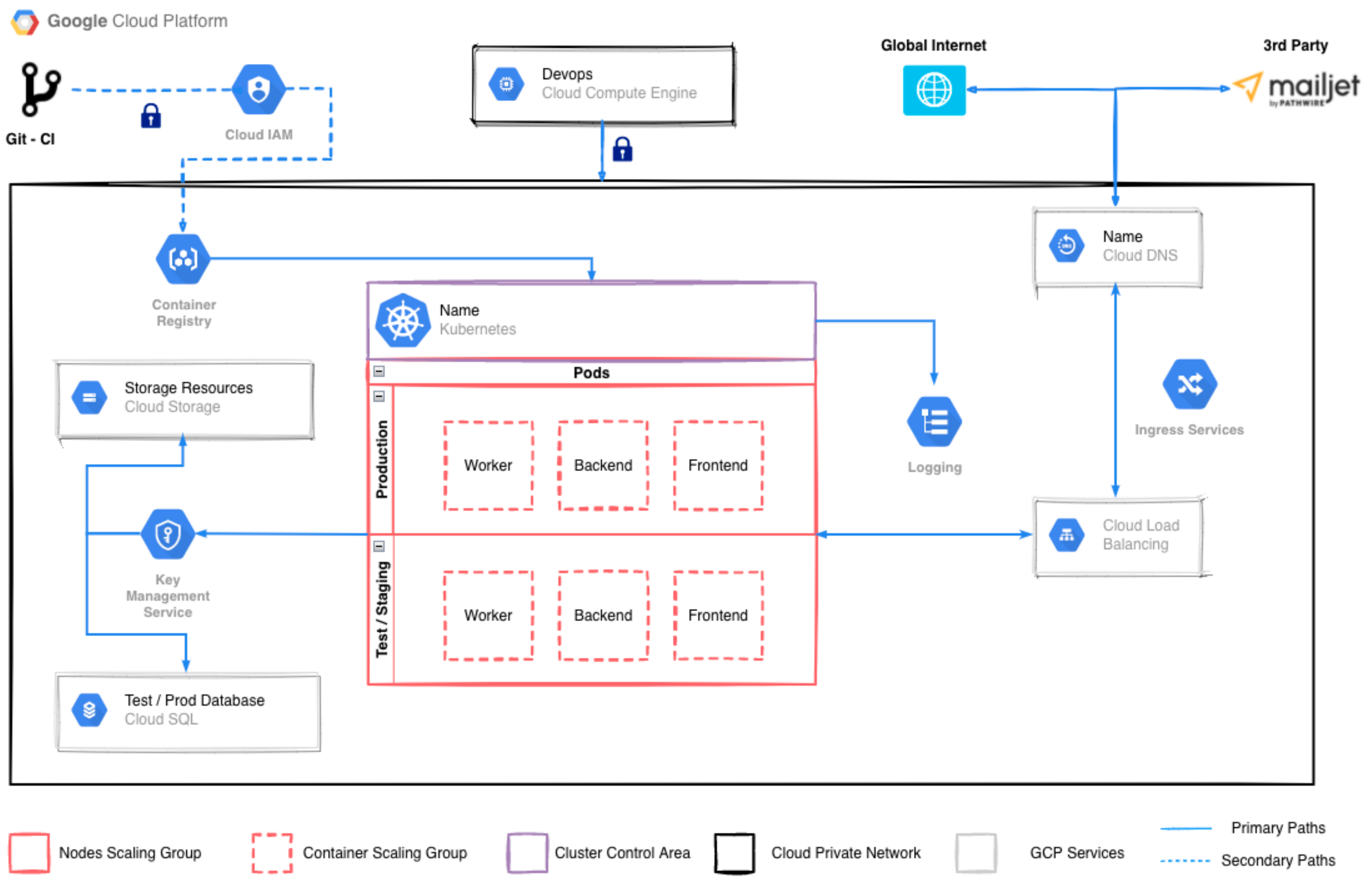
It is worth noting that the platform's performance can also be influenced by factors such as network connectivity, hardware capabilities, and user behavior. To mitigate these factors, the RUSH platform provides guidelines and best practices for users to optimise their own data handling processes and network connectivity.

Deployment Strategy

The RUSH platform follows a deployment strategy that leverages the capabilities of the Google Cloud Platform (GCP) to ensure efficient and reliable deployment of the application. The deployment strategy includes the use of Google Kubernetes Engine (GKE) to manage containers, the storage of container images in the Container Registry with git hash suffixes, the utilisation of ingress and load balancers for routing traffic, Cloud DNS for domain management, and IAM key management services for secure access to CloudSQL using gcloud proxy. Here's an explanation of each component of the deployment strategy:

1. **Google Kubernetes Engine (GKE):**
   - GKE is utilised as the container orchestration platform for deploying and managing the RUSH platform's containers.
   - The application is deployed in two clusters: the test cluster and the production cluster.
   - The test cluster receives updates from the main branch, allowing for continuous integration and testing of new features and code changes.
   - The production cluster receives tagged releases, ensuring stability and reliability for the live environment.
2. **Container Registry:**
   - Container images of the RUSH platform are stored in the Google Container Registry.
   - Each container image is suffixed with a git hash, providing a unique identifier for version control and traceability.
   - This approach allows for efficient image management, rollbacks, and reproducible deployments.
3. **Ingress, Load Balancers, and Cloud DNS:**
   - Ingress and load balancers are utilised to route and distribute traffic to the RUSH platform's services within the GKE clusters.
   - Ingress acts as the entry point, directing requests to the appropriate services based on defined rules.
   - Load balancers ensure high availability and scalability by distributing traffic across multiple instances of the platform.
   - Cloud DNS is used for domain management, mapping domain names to the respective IP addresses of the deployed services.
4. **CloudSQL and IAM Key Management Services:**
   - The RUSH platform accesses CloudSQL, the managed relational database service on GCP, for data storage and retrieval.

- IAM key management services are utilised to securely connect to CloudSQL using the gcloud proxy.
- This approach ensures secure and controlled access to the database, limiting exposure of sensitive information.



By utilising GCP services such as GKE, Container Registry, ingress, load balancers, Cloud DNS, CloudSQL, and IAM key management services, the RUSH platform benefits from a robust and scalable deployment strategy. It enables efficient management of containers, version control of images, routing and distribution of traffic, secure access to the database, and reliable domain management. This deployment strategy ensures a stable and performant environment for running the RUSH platform, facilitating seamless user access and interaction.

> To view the example deployment script for the RUSH platform, please refer to the following link: **RUSH Platform CI/CD**.

Testing Strategy

Testing Framework and Tools

The RUSH platform employs a comprehensive testing strategy to ensure the reliability, functionality, and quality of both its back-end and front-end components. The testing strategy encompasses different levels of testing, including back-end testing with Django Test, front-end testing with Jest, and container network testing with HTTP (bash). Here is an overview of the testing strategy for the RUSH platform:

Back-end Testing with Django Test

- The back-end testing of the RUSH platform is conducted using Django Test, a testing framework provided by Django.
- Django Test enables the creation of test cases and test suites to evaluate the functionality and behavior of the back-end components.
- Back-end testing focuses on unit tests, integration tests, and API tests to validate individual modules, their interactions, and the API endpoints.
- Test cases cover various scenarios, including positive and negative input cases, edge cases, and boundary conditions to ensure robustness and accuracy.

Front-end Testing with Jest

- The front-end testing of the RUSH platform is performed using Jest, a JavaScript testing framework widely used for testing React applications.
- Jest facilitates the creation of unit tests, integration tests, and component tests to assess the behavior and functionality of the front-end components.
- Front-end testing focuses on validating the UI components, user interactions, and the correctness of the application's logic and state management.
- Test cases cover various scenarios, including rendering components, user actions, and expected outcomes to ensure the desired user experience and functionality.

Container Network Testing with HTTP (bash) WILL BE REPLACED BY SELENIUM-HQ:

- The RUSH platform conducts container network testing using HTTP (bash) to assess the communication and network connectivity between different containers within the Docker environment.
- Container network testing ensures that the back-end, worker, and front-end containers can communicate effectively and exchange data seamlessly.
- Test scenarios involve sending HTTP requests and verifying the responses, ensuring the expected data flow and connectivity between containers.

The testing strategy for the RUSH platform aims to achieve thorough coverage across the back-end, front-end, and container network aspects. It focuses on validating the functionality, data flow, interactions, and network connectivity within the platform. Test cases are designed to cover a wide range of scenarios, including normal operation, edge cases, and potential error conditions.

Hardware Capability Evaluation

In addition to the testing strategies mentioned earlier, the RUSH platform recognise the importance of stress testing to evaluate the hardware capability and performance under heavy workloads. This specifically applies to resource-intensive tasks such as data validation and data seeding from the Excel bulk data upload feature. Stress testing is conducted to simulate high-demand scenarios and identify potential bottlenecks or performance issues. Here's an explanation of the stress testing approach:

Stress Testing

- Stress testing involves subjecting the RUSH platform to simulated high-volume and high-concurrency scenarios to evaluate its performance and robustness under heavy workloads.
- During stress testing, the platform is tested with large datasets or concurrent user loads that closely represent real-world usage scenarios.
- The focus is on measuring the response time, throughput, and resource utilisation to identify any performance degradation, scalability issues, or resource limitations.

Data Validation Stress Test

- A stress test specifically targeting the data validation process is conducted to assess how the platform performs when validating large volumes of data from the Excel bulk data upload feature.
- The stress test involves simulating multiple concurrent data uploads, each containing a significant amount of data that requires validation.
- The test measures the time taken to process and validate the data, ensuring that the platform maintains acceptable performance levels and does not become overwhelmed by the workload.

Data Seeding Stress Test

- A stress test focusing on the data seeding process is conducted to evaluate the platform's capability to handle heavy data seeding operations resulting from the Excel bulk data upload feature.
- The stress test involves simulating a high number of concurrent data seeding requests, each involving a large dataset to be inserted into the database.
- The test measures the time taken to seed the data, ensuring that the platform can handle the load without compromising performance or causing data integrity issues.

The stress testing process aims to identify any performance bottlenecks, resource limitations, or scalability issues that may arise when the platform is subjected to heavy workloads. By conducting stress tests, the development team can gather valuable insights and make necessary optimisations to ensure that the platform can handle the expected load and perform optimally under stressful conditions.

> The stress testing phase is important to validate the hardware capability and scalability of the RUSH platform, particularly during resource-intensive tasks like data validation and data seeding from the Excel bulk data upload feature.

Assumptions and Constraints

The development and operation of the RUSH platform are subject to certain assumptions and constraints that influence its design and functionality. These assumptions and constraints are important to consider as they provide context and boundaries for the platform's implementation. Here are the key assumptions and constraints of the RUSH platform:

Technical Infrastructure: The RUSH platform assumes access to a reliable technical infrastructure, including servers, networking components, and cloud-based services. It requires sufficient computational resources, storage capacity, and network connectivity to handle the expected user load and data processing requirements.

1. **Data Availability and Quality**: The platform assumes the availability and quality of data from various sources, including county and national levels. It relies on the assumption that relevant data is collected, validated, and provided by the respective stakeholders. The accuracy, completeness, and timeliness of the data are crucial for effective analysis and decision-making within the platform.
2. **Compliance with Regulatory Requirements**: The RUSH platform operates under the assumption that it complies with applicable laws, regulations, and data privacy requirements. It is assumed that necessary consent, data usage, and privacy policies are in place to protect user data and comply with legal obligations.
3. **User Adoption and Engagement**: The platform assumes user adoption and engagement, as its success relies on active participation and utilisation by relevant stakeholders. It assumes that users, including data entry staff, data approvers, administrators, and institutional users, will actively use the platform, contribute accurate data, and engage in data analysis and decision-making processes.
4. **System Scalability and Performance**: The RUSH platform assumes that it can scale and perform adequately to handle increasing user demand and growing data volumes over time. It assumes that the necessary infrastructure and optimisations can be implemented to maintain system performance, responsiveness, and reliability as the user base and data size expand.
5. **Collaboration and Data Sharing**: The platform assumes a collaborative environment and willingness among stakeholders to share data and insights. It assumes that relevant agencies, organisations, and institutions are willing to collaborate, contribute data, and use the platform's functionalities for informed decision-making and improved sanitation and hygiene practices.
6. **Resource Constraints**: The development and maintenance of the RUSH platform operate within resource constraints, such as budgetary limitations, time constraints, and

availability of skilled personnel. These constraints may impact the scope, timeline, and features of the platform's implementation and ongoing operations.

Dependencies

Software Dependencies

The RUSH platform incorporates various dependencies and frameworks to enable its functionality and deliver a seamless user experience. The following dependencies are essential components used in the development of the platform:

1. **Django:** The RUSH platform utilises Django, a high-level Python web framework, to build the back-end infrastructure. Django provides a solid foundation for handling data management, authentication, and implementing business logic.
2. **Pandas:** The platform relies on Pandas, a powerful data manipulation and analysis library in Python, to handle data processing tasks efficiently. Pandas enables tasks such as data filtering, transformation, and aggregation, enhancing the platform's data management capabilities.
3. **React:** The front-end of the RUSH platform is developed using React, a popular JavaScript library for building user interfaces. React enables the creation of dynamic and interactive UI components, ensuring a responsive and engaging user experience.
4. **Ant Design (antd)**: The platform utilises Ant Design, a comprehensive UI library based on React, to design and implement a consistent and visually appealing user interface. Ant Design provides a rich set of customisable and reusable UI components, streamlining the development process.
5. **Echarts:** Echarts, a powerful charting library, is integrated into the RUSH platform to generate various data visualisations. With Echarts, the platform can display charts, graphs, and other visual representations of data, enabling users to gain insights and make informed decisions.
6. **D3:** The RUSH platform incorporates D3.js, a JavaScript library for data visualisation. D3.js provides a powerful set of tools for creating interactive and customisable data visualisations, including charts, graphs, and other visual representations. By leveraging D3.js, the platform can deliver dynamic and engaging data visualisations to users.
7. **Leaflet:** The platform incorporates Leaflet, a JavaScript library for interactive maps, to handle geo-spatial data visualisation. Leaflet enables the integration of maps, markers, and other geo-spatial features, enhancing the platform's ability to represent and analise location-based information.
8. **Node-sass:** Node-sass is a Node.js library that enables the compilation of Sass (Syntactically Awesome Style Sheets) files into CSS. The RUSH platform uses node-sass to process and compile Sass files, allowing for a more efficient and maintainable approach to styling the user interface.

In addition to the previously mentioned dependencies, the RUSH platform relies on the following essential dependencies and libraries to support its functionality and development process:

1. **Django Rest Framework (DRF):** The RUSH platform utilises Django Rest Framework, a powerful and flexible toolkit for building Web APIs. DRF simplifies the development of APIs within the platform, providing features such as request/response handling, authentication, serialisation, and validation. It enables seamless integration of RESTful API endpoints, allowing for efficient communication between the frontend and backend components.
2. **PyJWT:** PyJWT is a Python library that enables the implementation of JSON Web Tokens (JWT) for secure user authentication and authorisation. The RUSH platform utilises PyJWT to generate, validate, and manage JWT tokens. JWT tokens play a crucial role in ensuring secure user sessions, granting authorised access to specific functionalities and data within the platform.
3. **Sphinx:** Sphinx is a documentation generation tool widely used in Python projects. The RUSH platform incorporates Sphinx to generate comprehensive and user-friendly documentation. Sphinx facilitates the creation of structured documentation, including API references, code examples, and user guides. It streamlines the documentation process, making it easier for developers and users to understand and utilise the platform's features and functionalities.

By leveraging these additional dependencies, including Django Rest Framework, PyJWT, and Sphinx, the RUSH platform gains essential support for building robust APIs, implementing secure authentication mechanisms, and generating comprehensive documentation.

These dependencies contribute to the platform's overall functionality, security, and user-friendliness, ensuring a well-rounded and effective solution for managing sanitation and hygiene practices in Kenya.

Master Lists

The RUSH platform incorporates several master lists that play a vital role in its functioning and data management. These master lists include the administrative levels, questionnaire definitions, and the shape-file representing accurate administrative boundaries. The administrative levels master list defines the hierarchical structure of Kenya's administrative divisions, facilitating data organisation, user roles, and reporting.

Shape-file and Country Administrative Description

An essential master list in the RUSH platform is the shape-file that accurately represents the administrative levels of Kenya. This shape-file serves as a crucial reference for various components within the system, including user management, data management, and visualisation. The importance of the shape-file as a master list lies in its ability to provide precise and standardised administrative boundaries, enabling effective data identification, filtering, and visualisation. Here's an explanation of the significance of the shape-file in the RUSH platform:

1. **Accurate Administrative Boundaries:**
   - The shape-file provides accurate and up-to-date administrative boundaries of Kenya, including the national, county, sub-county, and ward levels.
   - These boundaries define the jurisdictional divisions within the country and serve as a fundamental reference for assigning roles, managing data, and generating reports within the platform.
   - The accuracy of administrative boundaries ensures that data and administrative processes align with the established administrative hierarchy in Kenya.
2. **Data Identification and Filtering:**
   - The shape-file enables efficient data identification and filtering based on administrative boundaries.
   - By associating data points with the corresponding administrative levels, the platform can retrieve and present data specific to a particular county, sub-county, or ward.
   - This functionality allows users to view, analise, and report on data at different administrative levels, facilitating targeted decision-making and resource allocation.
3. **Visualisation and Geographic Context:**
   - The shape-file serves as the basis for visualising data on maps within the RUSH platform.
   - By overlaying data on the accurate administrative boundaries provided by the shapefile, users can visualise the distribution of sanitation and hygiene indicators across different regions of Kenya.
   - This geo-spatial visualisation enhances understanding, supports data-driven decision-making, and aids in identifying geographic patterns and disparities.
4. **Data Consistency and Standardisation:**
   - The shape-file, being a standardised and authoritative source, ensures consistency and uniformity in defining administrative boundaries across the platform.
   - It provides a reliable reference that aligns with the official administrative divisions recognised by the Ministry of Health and other relevant authorities.
   - The use of a consistent and standardised master list facilitates data aggregation, analysis, and reporting, ensuring reliable and comparable insights.

The shape-file sourced from the Ministry of Health should provide accurate administrative boundaries, supports data identification and filtering, enables geo-spatial visualisation, and ensures data consistency and standardisation. By utilising the shape-file as the master list, the platform can effectively manage administrative processes, present data in a meaningful geographic context, and contribute to evidence-based decision-making for improved sanitation and hygiene practices throughout Kenya.

> The shape-file sourced from the Ministry of Health acts as a crucial master list within the RUSH platform.

Questionnaire Definitions and Form Management

In addition to the administrative levels, the RUSH platform relies on another important master-list that defines the questionnaires used within the system. The questionnaire definition plays a crucial role in capturing the necessary data points and structuring the information collection process. Managing and maintaining the questionnaire forms are essential before seeding them into the system. This section outlines the importance of questionnaire definitions and the process of form management in the RUSH platform.

1. **Questionnaire Definitions:**
    - Questionnaire definitions define the structure, content, and data points to be collected during data entry.
    - These definitions specify the questions, response options, and any associated validations or skip patterns.
    - Questionnaire definitions determine the type and format of data that can be entered for each question.
    - These definitions ensure consistency and standardisation in data collection across the platform.
2. **Form Management:**
    - Form management involves the creation, customisation, and maintenance of the questionnaire forms.
    - Before seeding the forms into the system, it is crucial to ensure their accuracy, completeness, and adherence to data collection standards.
    - Form management includes activities such as form design, validation rules setup, skip logic configuration, and user interface customisation.
    - It is important to conduct thorough testing and quality assurance to ensure that the forms function correctly and capture the required data accurately.
3. **Form Fixes and Updates:**
    - As part of the form management process, it is essential to address any issues or errors identified during testing or from user feedback.
    - Form fixes and updates may involve resolving bugs, improving user interface elements, modifying question wording, or adjusting validation rules.
    - It is crucial to carefully test and validate the fixed forms to ensure that the changes are successfully implemented and do not introduce new issues.

> It is important to note that form management is an iterative process that may involve continuous improvements and updates as new requirements, feedback, or changes in data collection standards arise.

3rd-Party Services

The RUSH platform relies on certain third-party services to enhance its functionality and provide essential features. These services include **Mailjet** for email communication and optionally Cloud Bucket as a storage service. Here's an explanation of their significance:

1. **Mailjet**:
    - Mailjet is utilised for seamless email communication within the RUSH platform.
    - It provides features such as email delivery, tracking, and management, ensuring reliable and efficient communication between system users.

    - Mailjet enables the platform to send notifications, reports, and other email-based communications to users, enhancing user engagement and system responsiveness.
2. **Cloud Bucket** (Optional):
    - The RUSH platform offers the option to utilise Cloud Bucket, a cloud-based storage service, for storing data such as uploaded or downloaded Excel files.
    - Cloud Bucket provides a secure and scalable storage solution, allowing for efficient management of large data files.
    - By utilising Cloud Bucket, the platform offloads the burden of storing and managing data files from the host server, resulting in improved performance and scalability.
    - Storing data files in Cloud Bucket also enhances data availability, durability, and accessibility, ensuring seamless access to files across the platform.

> The use of Cloud Bucket as a storage service is optional, and alternative storage solutions can be considered based on specific requirements and constraints of the RUSH platform.

Risks and Mitigation Strategies

The development and operation of the RUSH platform come with inherent risks that can impact its effectiveness, security, and usability. Identifying and addressing these risks through appropriate mitigation strategies is essential to ensure the smooth functioning and success of the platform. Here are some key risks associated with the RUSH platform and their corresponding mitigation strategies:

Data Security and Privacy Risks

Risk: Unauthorised access, data breaches, or misuse of sensitive information.
Mitigation: Implement robust security measures, such as encryption, access controls, and regular security updates. Conduct thorough security audits, provide user education on data security best practices, and ensure compliance with data protection regulations.

Technical Risks

Risk: System failures, infrastructure disruptions, or performance bottlenecks.
Mitigation: Employ redundant and scalable infrastructure to minimise single points of failure. Regularly monitor system performance, conduct load testing, and implement disaster recovery plans. Update software and hardware components to address vulnerabilities and ensure optimal performance.

Data Quality Risks

Risk: Inaccurate, incomplete, or unreliable data affecting decision-making processes.
Mitigation: Implement data validation mechanisms, enforce data entry standards, and provide

user training on data collection best practices. Conduct regular data quality checks and provide feedback loops to data entry staff for improvement. Collaborate with data providers to improve data accuracy and completeness.

User Adoption and Engagement Risks

Risk: Low user adoption, resistance to change, or lack of engagement with the platform.
Mitigation: Conduct user needs assessments, involve stakeholders in the platform's design and development process, and provide comprehensive user training and support. Highlight the benefits and value of the platform to promote user adoption and engagement. Continuously gather user feedback and iterate on the platform based on user needs and preferences.

Stakeholder Collaboration Risks

Risk: Limited collaboration and data sharing among stakeholders.
Mitigation: Foster strong partnerships with relevant agencies, organisations, and institutions. Promote a culture of collaboration, sharing best practices, and jointly addressing common challenges. Establish clear data sharing agreements and protocols to encourage stakeholder participation and data contribution.

Resource Risks

Risk: Insufficient resources (human, or technical) for platform development and maintenance.
Mitigation: Develop realistic resource plans and secure adequate funding for the platform's implementation and ongoing operation. Optimise resource allocation, prioritise critical features and functionalities, and leverage partnerships to share resources and expertise.

> Regular risk assessments, monitoring, and proactive risk management practices should be integrated into the platform's lifecycle to identify emerging risks and implement appropriate mitigation strategies.

Implementation Plan

The implementation plan for the RUSH platform involves a structured approach to ensure successful development and deployment. The plan includes tasks, timelines, and resource requirements, taking into account the available team members. Here's an outline of the implementation plan:

Task Breakdown

1. Analise requirements and finalise specifications.
2. Design the system architecture and database schema.

3. Develop the back-end functionality, including data management, API integration, and authentication.
4. Implement the front-end components, including user interface design, data visualisation, and user interactions.
5. Integrate and test the front-end and back-end components for seamless functionality.
6. Implement security measures, including JWT authentication and secure data handling.
7. Conduct thorough testing, including unit tests, integration tests, and user acceptance testing.
8. Refine and optimise performance for data processing and visualisation.
9. Prepare documentation, including user guides, API documentation, and system architecture documentation.
10. Plan and execute the deployment strategy on the Google Cloud Platform.

Timelines

1. Analise requirements and finalise specifications: 1 week
2. System architecture and database schema design: 1 week
3. Back-end development: x weeks
4. Front-end development: x weeks
5. Integration and testing: x weeks
6. Security implementation: x weeks
7. Thorough testing and optimisation: x weeks
8. Documentation preparation: 1 week
9. Deployment on the Google Cloud Platform: 1 week

Resource Requirements

1. **2 Back-end Developers**: Responsible for back-end development, API integration, and database management.
2. **2 Front-end Engineers**: Responsible for front-end development, user interface design, and data visualisation.
3. **1 Project Supervisor**: Oversees the project, provides guidance, and ensures adherence to requirements, timelines and Pull Request reviews.
4. **1 Project Manager**: Manages the project's overall progress, coordinates resources, and communicates with stakeholders.
5. **1 Dev-ops Engineer**: Handles deployment, infrastructure setup, and configuration on the Google Cloud Platform.

The team members work collaboratively to ensure timely completion of tasks, quality assurance, and adherence to project milestones. Regular communication, coordination, and agile project management practices contribute to effective resource utilisation and smooth implementation.

> It is important to note that the timelines provided are estimates and can be adjusted based on the complexity of the project, team dynamics, and any unforeseen challenges that may arise during implementation.

Communication and Task Management

To facilitate efficient communication and task management within the team, the RUSH platform utilises **Slack** and **Asana**. These tools play crucial roles in enabling effective collaboration, communication, and task tracking.

Document Management

For document management, the RUSH platform utilises **Google Drive**. Team members can use Google Drive to store and manage various project documents, including design specifications, meeting minutes, reports, and other relevant files.

Report Hierarchy

The RUSH project follows a hierarchical reporting structure to ensure efficient communication and progress tracking. The hierarchy is designed to provide clear lines of reporting and facilitate effective decision-making. Here's an overview of the report hierarchy:

1. **Team Members**
    - Back-end Developers, Front-end Engineers, and Dev-ops Person directly report their progress, challenges, and updates to the Project Supervisor and Project Manager.
    - They communicate their completed tasks, pending work, and any obstacles they encounter during their development and deployment activities with Asana.
2. **Project Supervisor**
    - The Project Supervisor oversees the technical aspects of the project.
    - They provide guidance, support, and technical expertise to the team members.
    - The Project Supervisor break down all the tasks in Asana and assign to team members with due date.
    - The Project Supervisor works closely with the Project Manager to ensure alignment with project goals and timelines.
3. **Project Manager**
    - The Project Manager is responsible for the overall management and coordination of the RUSH project.
    - They track the progress of the development, monitor task completion, and manage resources and timelines.
    - The Project Manager communicates project updates, risks, and milestones to stakeholders and ensures effective collaboration among team members.

Regular meetings, such as stand-ups and sprint reviews, are conducted to discuss progress, address challenges, and align efforts across the team. This reporting hierarchy ensures effective communication, progress tracking, and efficient decision-making throughout the development and deployment phases of the RUSH platform.

Documentation References

The RUSH platform utilises various documentation references to provide comprehensive and accessible documentation for users and developers. These references include:

1. **Swagger:**
    - Swagger is used to generate interactive API documentation for the RUSH platform's Restful APIs.
    - By utilising the Open-API Specification, Swagger automatically generates detailed API documentation, including endpoint descriptions, request examples, and response details.
    - The Swagger documentation serves as a valuable resource for API consumers, facilitating seamless integration and understanding of the available endpoints and their functionality.
2. **GitHub Wiki:**
    - The RUSH platform leverages GitHub Wiki as a documentation reference for storing and presenting project-related information.
    - The GitHub Wiki provides a collaborative space for developers to create and maintain documentation directly within the project's repository.
    - It allows for the organisation of documentation pages, versioning, and easy navigation, ensuring that the latest project information is readily available to team members and contributors.
3. **DBDocs.io:**
    - DBDocs is utilised to generate comprehensive documentation for the RUSH platform's database schema and structure.
    - DBDocs automatically extracts information from the database and generates clear and well-structured documentation.
    - The DBDocs documentation serves as a valuable reference for understanding the database design, relationships, and entity attributes.
4. **ReadTheDocs:**
    - ReadTheDocs is employed to host and present user and developer documentation for the RUSH platform.
    - ReadTheDocs allows for the creation of user-friendly and searchable documentation, making it easy for users to find the information they need.
    - It provides a centralised location for storing and organising documentation, ensuring that both technical and non-technical users can access the necessary resources.

These documentation references, including Swagger, GitHub Wiki, DBDocs.io, and ReadTheDocs, play integral roles in providing comprehensive, organised, and accessible documentation for the RUSH platform. By utilising these resources, the platform ensures that users, developers, and API consumers have the necessary information to effectively utilise and contribute to the platform.

Conclusion

The development of the RUSH platform involves a comprehensive low-level design (LLD) that encompasses various aspects, including its purpose, functional overview, user roles, administrative levels, dependencies, security considerations, testing strategies, and deployment plan. Through meticulous planning and consideration of these factors, the RUSH platform aims to address sanitation and hygiene challenges in rural and urban areas of Kenya effectively.

The platform's purpose is to provide real-time monitoring, information aggregation, and data analysis to support decision-making and improve sanitation and hygiene practices. With its capabilities such as data visualisation, questionnaire management, and user role administration, the RUSH platform empowers stakeholders at different administrative levels to make informed decisions and take appropriate actions.

The LLD also highlights the importance of master lists, including administrative levels and questionnaire definitions, which serve as crucial references for data management, user roles, and system operations. Additionally, the security considerations, testing strategies, and dependency management outlined in the LLD ensure robustness, performance, and reliability of the platform.

The deployment strategy leverages Google Cloud Platform, utilising containerisation with GKE, storing container images in the Container Registry, and employing services like CloudSQL, Cloud Storage Bucket, Ingress, Load Balancers, and Cloud DNS. The implementation plan provides a timeline, task breakdown, and resource requirements, allowing for efficient coordination and progress tracking.

Furthermore, the RUSH platform embraces effective communication and task management through the use of Slack and Asana, enabling seamless collaboration and efficient project execution. The documentation references, including Swagger, GitHub Wiki, DBDocs, and ReadTheDocs, facilitate comprehensive documentation and knowledge sharing among the team.

In conclusion, the RUSH platform's LLD serves as a foundational guide for its development, emphasise the importance of functionality, data management, security, testing, deployment, communication, and documentation. By adhering to this comprehensive design, the RUSH platform aims to make significant contributions to improving sanitation and hygiene practices, ultimately leading to better health outcomes in rural and urban areas of Kenya.

Revision #51
Created 30 May 2023 05:29:01 by Deden Bangkit
Updated 31 May 2023 16:13:17 by Deden Bangkit