Project name	WAS - Workflow Automation System

Project author		
№	Names of Participants	FN
1	Petar Petrov	5MI0800197

1. Short project description (Business needs and system features)

Workflow Automation System (WAS) is a web application which allows for users to not have the need to manually transfer data between different systems and third-party apps they are using. It provides the ability for a user to register and create custom workflows which listen and act on events coming from authenticated APIs. These created workflows also can be shared publicly to the rest of the users.

The main user roles will be:

- *Anonymous User* – can only view publicly shared workflows but can't create such.
- *Registered User* – can also create and share workflows.
- *Administrator* (extends *Registered User*) – can manage (create, edit user data and delete) all *Registered Users*, as well as arbitrary user workflows.

The integrations that will be supported:

Gmail, Github, *Slack*

The system will be developed using **Golang, Gin** OR **chi**, and **PostgreSQL** for the backend and **React.js + Typescript** for the frontend (using the **React Flow** library for the workflow editing and visualization)

The application will utilize the microservice architecture.

Each third-party integration will be separated into a microservice. Every workflow will consist of a set of **Workflow Elements**, which will be connected in the form of a Directed Acyclical Graph (DAG). Every Workflow Element can be of the following type:

- **Listener** - will listen for specific updates from the configured third-party API (using **Polling** OR **Webhooks**) e.g. receiving an email
- **Action** - after receiving an event from the Listener, the **Workflow Handler** forwards the data to the microservice for the corresponding party specified in the workflow which executes the specified action.
- **Utility** - this is a helpful element in the workflow which will transform incoming data from the Listener or from another workflow element and forward it further down the pipeline. Some examples are: filtering out data by a user-specified conditional (e.g. filter out incoming spam emails) or use parallelisation and send the output data from a workflow element to more than one Action.

For each third-party integration and for each type of Workflow Element, there will be a defined **Schema** (**JSON/OpenAPI**) which specifies for each element what type it is, what input type it expects and what output it produces. This will allow for the system to be easily extensible by adding multiple microservices and, potentially, also MCP services.

Additionally, there will be microservices for the **Workflow Handler** which handles the data for all of the ongoing workflows and their progress, an **Identity** microservice for storing the **User Data and sessions** and a microservice for the **Posts/Shared workflows** with the corresponding comments and reactions. All of these services will communicate with **gRPC** using **Protobuf** serialization.

For each *Gmail* Action, the user will be allowed to map incoming object/JSON fields to placeholders in an **Email template** (or if the Action is a *Slack* integration - a specific **Channel Message template**) through a helpful UI element.

To set up a workflow, the user needs to authenticate with the third-party APIs using **OAuth2**.

The client will communicate with the application server using **JSON** in a **REST API**.

2. Main Use Cases / Scenarios

Use case name	Brief Descriptions	Actors Involved
2.1. Browse available workflows	Every user can browse the publicly shared workflows in *WAS.* The display of the workflows will be paginated.	All users
2.2. Register	*Anonymous User* can register in the system by providing a valid e-mail address, first and last name, and choosing a password.	*Anonymous User*
2.3. View/Change User Data	*Registered User* can view the personal user data of all users but only edit their own. *Administrator* can view and edit the user data of all *Registered Users*.	*Registered User, Administrator*
2.4. Manage Users	*Administrator* can delete the account of a *Registered User* and also has the ability to make another user an *Administrator.*	*Administrator*
2.5. Manage a workflow	*Registered User* can create a new workflow and edit it later if needed. *Administrator* can view all available user workflows and delete any of them.	*Registered User, Administrator*
2.6. Activate a workflow	After creating a workflow, *Registered User* can activate it to listen for events.	*Registered User*
2.7. Share workflow	*Registered User* has the option to share their workflow as a post to the public to view and reuse.	*Registered User*

2.8. View posts	*Registered User* can view publicly shared workflows.	*Registered User*
2.9. Comment/react to posts	*Registered User* can comment/react to each publicly shared workflow/post.	*Registered User*

3. Main Views (Frontend)		
View name	**Brief Descriptions**	**URI**
3.1. Home	Presents the introductory information for the purpose of the system as well as detailed instructions on how to start using it. Prominently offers ability to register.	/
3.2. Workflows	Presents the available workflows to *Registered User* and also separates them depending on if they are active or not. Additionally, the ability to edit or create a new workflow is presented here.	*/workflows*
3.3. All workflows	Presents all of the workflows in the app in a paginated table with the ability to delete each one. This is only available for *Administrator.*	*/all-workflows*
3.4. All users	Presents all of the users in the app in a paginated table with the ability to delete each one or promote each one to *Administrator*.	*/all-users*
3.5. User Registration	Presents a view allowing the *Anonymous Users* to register in *WASA*	*/register*
3.6. Login	Presents a view allowing the users to login.	*/login*
3.7. Profile	Provides ability to view and edit personal *User Data* such as first name, last name or email.	*/personal*

3.8. Workflow editor	Presents the ability to edit a current workflow or work on a newly created one. Also presents the ability to activate or share the current workflow.	*/workflow*
3.9 Shared workflows	Presents to the *Registered user* all of the publicly shared posts/workflows and the ability to react/comment on each one of them.	*/posts*
3.10. About	Presents information about the *WAS* project and his owner.	*/about*

4. API Resources (Backend)

View name	**Brief Descriptions**	**URI**
4.1. Users	GET *User Data* for all users. Available only for *Administrators*.	*/api/users*
4.2. User	GET, PUT, DELETE *User Data* for *User* with specified *userId*, according to restrictions described in UCs.	*/api/users/{userId}*
4.3. Login	POST *User Credentials* (e-mail address and password) and receive a valid *Session Token/JWT* to use in subsequent API requests.	*/api/login*
4.4. Logout	POST a logout request for ending the active session with *WAS,* and invalidating the issued *Session token/JWT*.	*/api/logout*
4.5. Workflows	GET all available workflows. Available only for *Administrator.* POST to create a new workflow.	*/api/workflows*
4.6. Workflow	GET, PUT, DELETE Workflow for a given *Registered User*.	*/api/workflows/{workflowId}*
4.7. Posts	GET all public posts/shared workflows. POST to create a new post/share a workflow.	*/api/posts*

4.8. Post	GET, DELETE a post.	*/api/post/{postId}*
4.9 Schema	GET the schema for a specific type of Workflow Element and specific third-party integration.	*/api/schema*
4.10 Email Templates	GET the email template for a specific Workflow Element which is a *Gmail* integration	*/api/email-template/{workflowElementId}*
4.11 Channel Message Template	GET the channel message template for a specific Workflow Element which is a *Slack* integration	*/api/channel-message-template/{workflowElementId}*