Gatino - Innovation Gateway for Swinburne Vietnam

System Requirements Specification

GROUP 1

Name	Position	Email	Phone
Ta Quang Tung	Team Leader, Client Liaison	104222196@student.swin.edu.au	0921426803
Nguyen Quang Huy	Development Manager	104169507@student.swin.edu.au	0968751524
Tran Hoang Hai Anh	Support Manager	104177513@student.swin.edu.au	0866899518
Phan Sy Tuan	Quality Manager	104072029@student.swin.edu.au	0888052003
Duong Quang Thanh	Planning Manager	104167828@student.swin.edu.au	0976663084

COS40006, COMPUTING TECHNOLOGY PROJECT B, SPRING 2025

Document Change Control

Version	Date	Authors	Summary of Changes
1.0	6/10/2024	Ta Quang Tung, Nguyen Quang Huy, Tran Hoang Hai Anh, Phan Sy Tuan, Duong Quang Thanh	Initial version of the document.
2.0	6/4/2025	Ta Quang Tung, Duong Quang Thanh	Updated the document to reflect changes in CTPB.

Document Sign Off

Name	Position	Signature	Date
Ta Quang Tung	Team Leader, Client Liaison	Quang Tung Ta	6/4/2025
Nguyen Quang Huy	Development Manager	Quang Huy Nguyen	6/4/2025
Tran Hoang Hai Anh	Support Manager	Hoang Hai Anh Tran	6/4/2025
Phan Sy Tuan	Quality Manager	Sy Tuan Phan	6/4/2025
Duong Quang Thanh	Planning Manager	Quang Thanh Duong	6/4/2025

Client Sign Off

Name	Position	Signature	Date
Dr. Le Minh Duc	Head of IT Department, Swinburne Vietnam Head of Innovation Lab, Swinburne Vietnam	Le, Minh Duc	6/4/2025
Organisation: Innovation Lab, Department of Computer Science, Swinburne Vietnam			

1 - Introduction

Gatino - Innovation Gateway is a platform designed for Swinburne Vietnam to facilitate the organization, management, and demonstration of student innovation projects. This platform aims to streamline the discovery and interaction with student projects across a variety of disciplines, enhancing accessibility and engagement within Swinburne's academic community.

1.1. - Purpose

The purpose of this System Requirements Specification (SRS) is to outline the necessary functional and nonfunctional requirements to guide the development team in building a system that meets the needs of Swinburne University Vietnam's Innovation Lab. This document is intended for use by project developers, system architects, quality assurance teams, and stakeholders, including Swinburne's IT department and core Gatino team, as a reference to ensure that all requirements and constraints are addressed throughout the development lifecycle.

1.2. - Scope

Name of the system: Gatino - Innovation Gateway for Swinburne Vietnam

Boundary of the project: Within the scope of this project, we aim to fully develop the search feature and demonstration feature of Gatino.

What the project will accomplish:

- Create a search engine capable of finding projects based on various metadata (name, description, technology used), source code, and associated documentation.
- Research and implement ways for users to interact with app demonstrations via the browser, offering three interaction modes: constrained (partial) live, and fully live, depending on the project's constraints.
- Package the search and demo functionalities as independent services that can be integrated into other Gatino components.

What the project will not accomplish:

- Enable the demonstration of products **outside** the following categories: web, mobile (Android), IoT.
- Implement the capability to deploy innovation projects.
- Provide ongoing system maintenance after deployment.

Application and benefits of the system:

The system will centralize project management and demonstration, allowing users to easily search for and interact with student innovation projects. Its responsive interface, robust

search engine, and queue system for demos will enhance user experience. By packaging features as independent services, the system ensures flexibility, scalability, and future integration with other Gatino components.

1.3. - Definitions, Acronyms and Abbreviations

Acronym/Abbreviations	**Meaning**
SRS	Software Requirements Specification
IDE	Integrated Development Environment
UI	User Interface
GUI	Graphical User Interface
IoT	Internet of Things
AI	Artificial Intelligence

2 - Overall Description

In this project, we will develop the search, demonstration (demo) features, and product management for dedicated roles of the Gatino platform. The search feature enables users to easily and quickly search for products, the demo feature lets them see and interact with a functional demonstration of innovation products, and the product management feature allows admins to manage the deployment of innovation products.

- **Type**: Part of the main system of the Gatino platform.
- **Purpose**: Provide a project search capability, let users view and interact with real project demonstrations.
- **Role**: Integral to the platform, aiding in project discovery and showcase.

2.1. - Product Features

The significant features of the software include:

- **Search Engine**: Advanced search functionality that allows searching not only based on the project titles but also on the descriptions and associated documents.
- **Search Information Extraction**: Extraction of text from various document formats (PDF, DOCX) and applied a pipeline to enhance search capabilities, as well as extraction of other technical metadata related to the project.
- **Project Demonstrations**: Interactive sessions with real, operational project demonstrations.
- **Project Management**: Manage the status of project, dashboard gives insight of technologies and coding language.

2.2. - System Complexity

The system as a whole, as well as the planned features, comes with the following challenges:

1. **Overall system**:
    - As the system will be integrated into the larger Gatino platform, it must be compatible with the technology stack used by the other Gatino team.
    - The system brings together many technologies: Docker, GitHub API, Apache Tika, Docker Android, etc.
2. **Search feature**:
    - Innovation projects come with a lot of information that will not be provided explicitly by the user but must be extracted to make the search more effective.

- Innovation projects can have a lot of searchable information, and thus removing redundant data to optimize storage is necessary.
- Innovation projects have many fields that can be searched over. Optimizing the search query such that it can search over all necessary fields while maintaining acceptable performance is challenging.

3. **Demo feature**:
    - Different types of projects have different levels of support for view, control, and concurrency capabilities. For example, a web project can be viewed and controlled by many users at the same time. On the other hand, an Android project may only be viewed and controlled by one user at a time.
    - Innovation projects use many different technologies and developing a common platform that supports all of them is a challenge. For example, one web project may use the MERN stack, while another may use Laravel or Ruby on Rails.

2.3. - System Requirements

The system will be deployed to a server managed by the Innovation Lab at Swinburne Vietnam, both during development and production. The purpose of deploying during development is to confirm that the developed features successfully integrate with the whole Gatino system. However, during this stage, the team can also run locally on their machines.

Development Environment

- **Software Requirements**:
    - Docker Engine.
    - Container images of the services used, including databases.
    - Operating System: Linux-based systems (Ubuntu recommended) or Windows with WSL (Windows Subsystem for Linux).
- **Hardware Requirements**:
    - Minimum 8 GB RAM.
    - At least 100 GB of disk space.

Production Environment

- **Software Requirements:**
    - Docker Engine.
    - Container images of the services used, including databases.
    - Domain name of the Gatino platform.

- **Hardware Requirements:**
    - Recommended 32 GB RAM or more.
    - SSD storage with at least 200 GB free space.
    - Server with a high-speed internet connection.

2.4. - Acceptance Criteria

The most basic acceptance criteria that the project must meet is that ***it must be verified to meet all of the requirements expressed in section 3, 4, and 5.*** If a criterion is revealed to be unsatisfiable during research and development, it must be discussed with the client for adjustment. In addition, the following high-level criteria must also be met for the system to be accepted:

Functional Criteria:

1. The Search Engine should **accurately** retrieve relevant projects based on the latest project information. The results should be sorted in **descending order of relevance**.
2. The Search Information Extraction module should be able to extract content from **a variety of file types**: PDFs, DOCXs, Markdown, OTFs.
3. Users should be able to run and interact with demonstrations of various project types without conflict with other users.
4. As far as the project supports it, the running demonstration should be as **real-time** as possible.

Performance Criteria:

1. The system must handle at least **100 simultaneous users** without performance degradation.
2. Search results should be returned in **less than 1 second**.
3. After a project has been uploaded or modified on the platform, the search feature should reflect the new information after **at most 5 minutes**.

Usability Criteria:

1. The user interface should be friendly and easy to get the most relevant results after searching.
2. The results should be presented in an easily navigable manner.
3. The search feature should be immediately usable for users who are familiar with Google Search. Otherwise, they should be able to figure out the Search feature in **less than 2 minutes**.

Scalability Criteria:

1. The system must support easy scaling using Docker without significant downtime.
2. The system is expected to support **around 100 projects** in its **first 2 years** of operation.

2.5. - Documentation

The following documents will be included in the project submission:

- Detailed architecture design diagram(s) of the developed features.
- Configuration and deployment instructions to set up the developed features on the server.
- Result of test cases and user acceptance tests.

3 - Functional Requirements

The functional requirements are separated into two work areas corresponding to the two features we plan to develop. They are framed with the Task and Support method (Lauesen, 2001).

Work area 1 - Search for projects

General	Search is an important feature of the system as it allows users to discover projects on the platform, a precondition to several other features such as demonstration or integration. Search should allow complex filters to refine the results while also maintaining simplicity. Ideally, it should function similarly to popular search engines such as Google, allowing the user to enter an easy-to-write search query and specify additional search parameters. The search module must also add/update/delete a project's search information when it is added/updated/deleted. A project's search information is not only given explicitly by the student who uploads it but can also be extracted from various sources, such as any provided links and documents.
Actors involved	**1 - Swinburne students** Students upload/update their projects and expect their project information to be searchable as quickly as possible. In addition, students may also want to search for projects from other students, either for reference purposes or for integration into their own projects. This group typically have medium-to-high technical experience and they may want to search by feature, technology and language used, and API integration support. **2 - Swinburne lecturers/Heads of Department** Lecturers want to search for specific student projects to demonstrate in lectures. Typically, they will search by project name or project type. Heads of Department want to use the search feature to better understand the most common technology/language choices for a given type of project. They are also interested in seeing how many projects have been developed for each major. While this makes more sense as a separate visualization feature, the search feature can report some basic statistics for each search query to fulfil this need. This user group have high technical experience. **3 - Businesses**

	Businesses may partner with Swinburne Vietnam to purchase student's software for their companies. To do this, they need to use the search feature. This group has varying levels of technical expertise. Some business partners are not familiar with technical terms and will only search by project feature, often expressed as keywords. Others have more technical knowledge and will want to search by tech stack or programming language used.

Data model

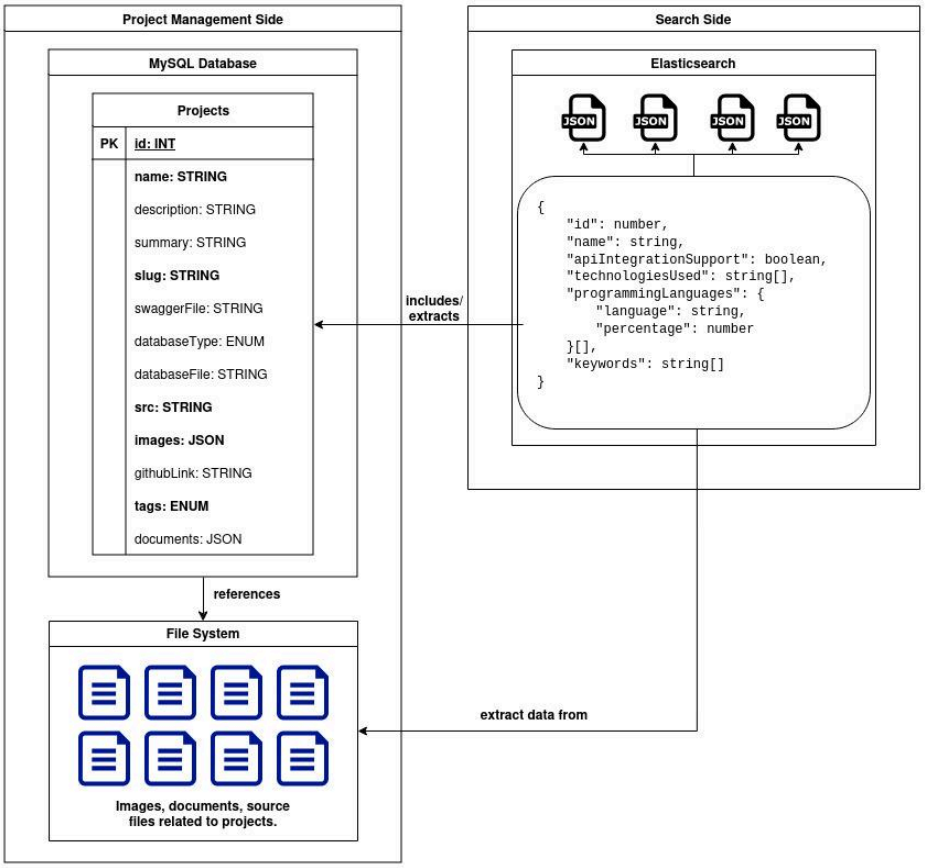

Fig. 1: The data model for the search feature. The full-resolution image can be seen here.

Project information will be stored in two places: the Project Management Side and the Search Side. The Project Management Side stores the project data that can be seen and directly modified by the user. Here, the data includes the project's metadata in a MySQL table and any files associated with the project in the server's file system.

The Search Side stores the project search information directly inside Elasticsearch as JSON documents. Every project has one associated document in Elasticsearch. This search data is built on the metadata stored in MySQL and the metadata extracted from project files and links. The system only searches on the Elasticsearch data, but it may ask the Management Module for additional project data when it returns the result to the user.

Task 1.1. - Search for projects

Purpose	Allow the user to search for student projects that have been uploaded to Gatino.
Trigger/precondition	A user opens the Projects page and initiates a search.
Frequency	10 searches/minute.
Critical	During a demonstration event where the audience members may want to search for a project simultaneously.

Sub-tasks	**Example solution**
1 - Provide search query and parameters. **Problems:** ● Users are not aware of the search parameters. ● The search query can represent one or more fields. ● The user may enter keywords that are synonymous to the ones in the database.	The system displays a search bar along with search parameters in the form of dropdowns checkboxes.
2 - Find projects that match the query and parameters. **Problem:** ● Some projects may not have been indexed and thus are unsearchable.	The system scans the database for projects that match the search query and returns the results in pages.
Variants	**Example solution**
1a - The user provides no search query or parameters.	Returns a list of all projects, sorted by date uploaded.
2a - No projects match the search query.	Informs the user that no projects match their search query, or shows them a list of random projects.

Workflow:

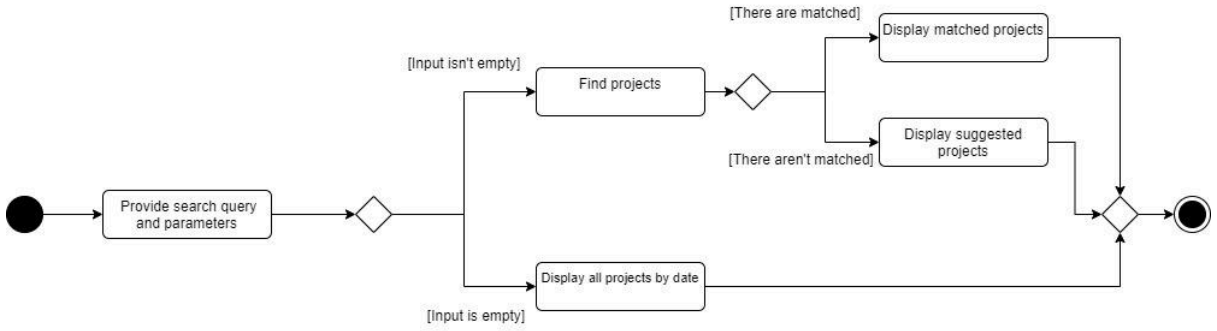

Fig 2: The workflow for task 1.1 - Search for projects

Task 1.2. - Add project search information

Purpose	Make a project searchable by adding its search information to the database.
Trigger/precondition	A student uploads a project to the platform. The upload includes basic details about the project (such as name and description) as well as files and links through which more search data can be extracted.
Frequency	10 projects/month.
Critical	● When Gatino is first launched, a large number of projects will be uploaded as students and lecturers migrate from existing platforms like Canvas. ● The search module may not be informed of the project upload due to network errors, etc.

Sub-tasks	**Example solution**
1 - Extract search information from the project. **Problems:** ● Search information can come from a variety of sources, such as the project's GitHub repository or uploaded documents. ● The uploaded documents may contain too much text.	The system extracts more information from the provided links and files using a variety of tools/services such as the GitHub API or Apache Tika. The system may also filter the data or transform it to reduce the amount of storage needed.
2 - Organize the extracted data into a common structure. **Problem:** The structure of the information may change in the future.	The system defines a common structure for the search data (JSON or YAML, etc.), including all the fields that will be included, then puts the obtained data into that structure.
3 - Save the structured search information. **Problem:** The search information may have to be saved separately from the main database, and the two databases have to be synchronized.	The system adds the organized project data into the search database.

Variants	Example solution
None.	

Workflow:

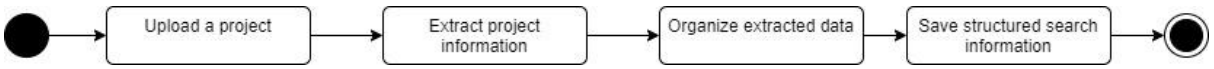

Fig 3: The workflow for task 1.2. - Add project search information

Task 1.3. - Update project search information

Purpose	Reflect the latest information about the project after it has been changed.
Trigger/precondition	A student updates a project that they have uploaded. The update may change the basic details about the project (such as name and description) as well as files and links through which more search data can be extracted.
Frequency	5 updates/month.
Critical	The search module may not be informed of the project update due to network errors, etc.

Sub-tasks	**Example solution**
1 - Extract search information from the updated project. **Problems:** ● Search information can come from a variety of sources, such as the project's GitHub repository or uploaded documents. ● The uploaded documents may contain too much text.	The system extracts updated information from the provided links and files using a variety of tools/services such as the GitHub API or Apache Tika. The system may also filter the data or transform it to reduce the amount of storage needed.
2 - Organize the extracted data into a common structure. **Problem:** The structure of the information may change in the future.	The system defines a common structure for the search data (JSON or YAML, etc.), including all the fields that will be included, then puts the obtained data into that structure.
3 - Save the updated search information. **Problems:** ● Some of the old search information may have to be deleted. ● The search information may have to be saved separately from the main	The system updates the search information in the database, removing any old data as necessary.

database, and the two databases have to be synchronized.	
Variants	**Example solution**
None.	

Workflow:

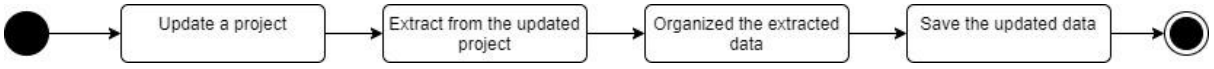

Fig 4: The workflow for task 1.3. - Update project search information

Task 1.4. - Delete project search information

Purpose	Stop a deleted project from being searchable.
Trigger/precondition	A student or admin archives or deletes a project.
Frequency	Very rarely, 5 projects/year.
Critical	The search module may not be informed of the project delete due to network errors, etc.

Sub-tasks	**Example solution**
1 - Erase the project's search information.	The system sends a query to delete the project information from the search database.
2 - Refresh the search engine to reflect the changes.	The system sends a query to refresh the search engine if necessary, or the engine refreshes itself.
Variants	**Example solution**
1a - The project has only been archived (soft-deleted) but not permanently deleted.	The system sets a boolean flag in the project's search information to make the engine ignore it.

Workflow:

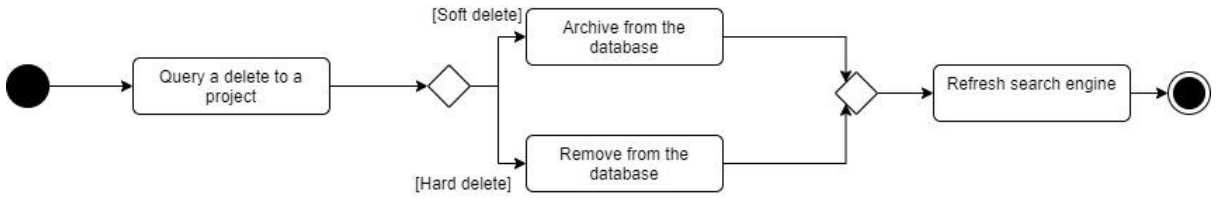

Fig 5: Task 1.4 - The workflow for task 1.4. - Delete project search information

Work area 2 - Demonstrate projects

General	Demonstration is a key feature of the platform as it enables users to run and interact with a project instead of just viewing its static information, helping them better understand its features and capabilities. Depending on the nature of the project, it may be partially or fully demonstrable. In addition, due to the lack of hardware/software resources, some projects can only be run by one user at a time. Projects can fall into many categories, although this project covers the following: ● **Web**: Web projects are typically fully viewable and interactive. Multiple users can access a web project at the same time with minimal interference. ● **Android (mobile):** Android projects are run on either an Android emulator or a physical device, the former being easier and more convenient to implement. Only one person can control the same emulator or physical device at a given time. These projects are fully viewable and interactive. ● **IoT**: These projects may or may not run on Gatino's server and often make heavy use of external APIs. In addition, some projects may involve physical sensors that are not controlled by Gatino. It is quite rare for these projects to have a user interface, unless they are part of a bigger web or Android project (i.e. a smart home app). Typically, these projects expose APIs that produce data (either as a continuous stream or on demand) and can be visualized.
Actors involved	All user groups, from students to lecturers and businesses, will want to access this feature. Their technical expertise ranges from very low to very high. They all expect to be able to view and interact with the demonstration as quickly as possible, and some will not tolerate any waiting time.

Data model

With regards to the Demo feature, the system does not store much information related to the projects. In addition, different project types will have the system concerned with different types of information. More specifically:

- For **web** projects, the system stores their source code and deploys them as Docker containers that can be turned on or off.
- For **Android** projects, the system uploads their APK files to a physical Android phone.

- For **IoT** projects, the system stores their source code and, depending on the configuration of the uploader, deploys them as Docker containers that can be turned on or off. Some projects may run outside of the Gatino platform. These projects may expose data-producing APIs that Gatino can use to build visualizations for the user interface.

Task 2.1. - Run a demonstration of a web project

Purpose	Enables a user to view and interact with a web project in operation.
Trigger/precondition	A user clicks on the “Run demo” button in the web project’s details page.
Frequency	20 concurrent demos/hour.
Critical	During an important showcase event (such as a lecture or Swinburne Experience Day), a large audience (assumed to be about 50-75) may want to run demos simultaneously.

Sub-tasks	**Example solution**
1 - Start the project on the server (optional). **Problem:** A project may take a few minutes to start.	The server issues a command to switch on the project’s Docker container when a demo is requested. Ahead of an important event, the system can allow the system admin to switch the project on in advance, saving time.
2 - Show the interactive, running project.	The server presents a web-based user interface through which the user can view the demonstration in action. The user can interact with the web interface to interact with the project itself.
3 - Stop the project on the server (optional). **Problem:** Other users may want to run the project demonstration after the first one.	The server issues a command to switch off the project’s Docker container. To be available for near future users, the container can stay on for a given period of time before being automatically shut off to save resources.

Variants	**Example solution**
None	

Workflow:

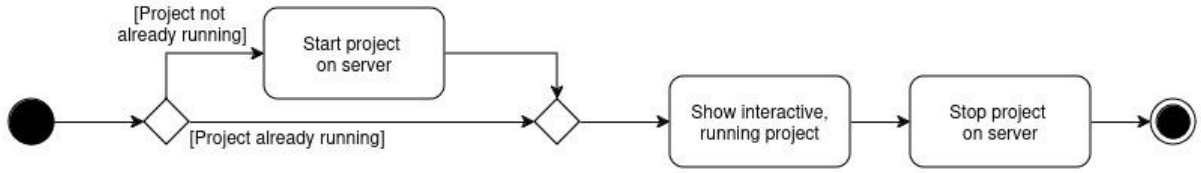

Fig 6: The workflow for task 2.1. - Run a demonstration of a web project

Task 2.2. - Run a demonstration of an Android project on a physical device

Purpose	Enables a user to view and interact with an Android project in operation.
Trigger/precondition	A user clicks on the “Run demo” button in the Android project’s details page.
Frequency	20 concurrent demos/hour.
Critical	During an important showcase event (such as a lecture or Swinburne Experience Day), a large audience (assumed to be about 50-75) may want to run demos simultaneously.

Sub-tasks	**Example solution**
1 - Start the Android phone on the server (optional). **Problem:** The phone may take a few minutes to start. One phone might not be enough if there are too many demo requests.	Ahead of an important event, the system admin switches the phone on. This phone cannot be turned on remotely.
2 - Show the interactive, running project.	The server presents a web-based user interface through which the user can view the demonstration in action. The user can interact with the web interface to interact with the project itself.
3 - Stop the Android phone on the server (optional). **Problem:** Other users may want to run the project demonstration after the first one.	To be available for near future users, the phone can stay on for a given period of time before being automatically shut off to save resources.
Variants	**Example solution**
2a - Another user is already accessing the phone.	The system places incoming users into a queue so that they can wait for their turn. The system will inform the user of their position in their queue and the estimated waiting time. To ensure fairness, allow only

	a few minutes for each user.

Workflow:

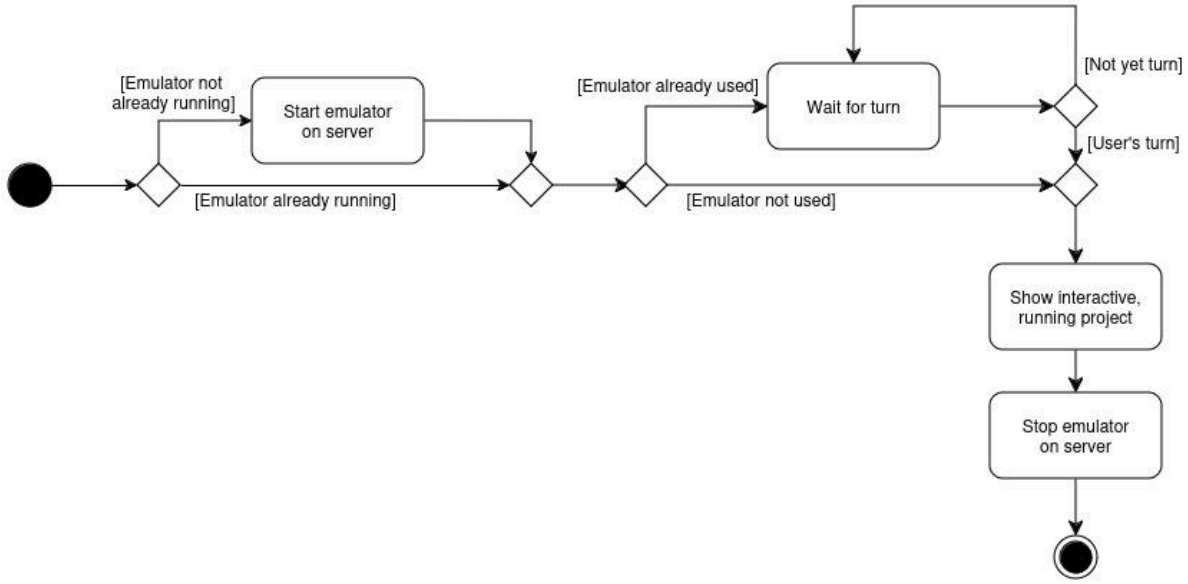

Fig 7: The workflow for task 2.2. - Run a demonstration of an Android project

Task 2.3. - Run a demonstration of an IoT project

Purpose	Enables a user to view and possibly interact with an IoT project in operation.
Trigger/precondition	A user clicks on the “Run demo” button in the IoT project’s details page.
Frequency	20 concurrent demos/hour.
Critical	During an important showcase event (such as a lecture or Swinburne Experience Day), a large audience (assumed to be about 50-75) may want to run demos simultaneously.

Sub-tasks	**Example solution**
1 - Start the project on the server (optional). **Problems:** A project may take a few minutes to start. Some projects may not run on the server and thus cannot be directly controlled.	The server issues a command to switch on the project’s Docker container when a demo is requested. Ahead of an important event, the system can allow the system admin to switch the project on in advance, saving time.
2 - Show the running (and possibly interactive) project. **Problems:** Some projects can only be controlled by one person at a time or may not be controllable at all. Some projects do not come with a user interface.	The server presents a web-based user interface through which the user can view the demonstration in action. The user may interact with the web interface to interact with the project itself.
3 - Stop the project on the server (optional). **Problem:** Other users may want to run the project demonstration after the first one.	The server issues a command to switch off the project’s Docker container. To be available for near future users, the container can stay on for a given period of time before being automatically shut off to save resources.
Variants	**Example solution**
2a - The demonstration can only be controlled by one user and cannot be replicated.	The system places incoming users into a queue so that they can wait for their turn. The system will inform the user of their position in their queue and the estimated

	waiting time. To ensure fairness, allow only a few minutes for each user.
2b - Some projects cannot be interacted with completely or at all.	The system presents as much information as it can show and allows as much interaction as it can support. In cases where a project cannot be interacted with at all, the system can present the user with a fallback video showing the developers demonstrating the project.
2c - Some projects do not have a built-in user interface.	If the project exposes, or links to, a data-producing API, the system can query the API for data to construct a visualization for the user. This visualization can be data charts or live video/image feeds, depending on the type of the project. In cases where a project cannot be viewed at all, the system can present the user with a fallback video showing the developers demonstrating the project.

Workflow:

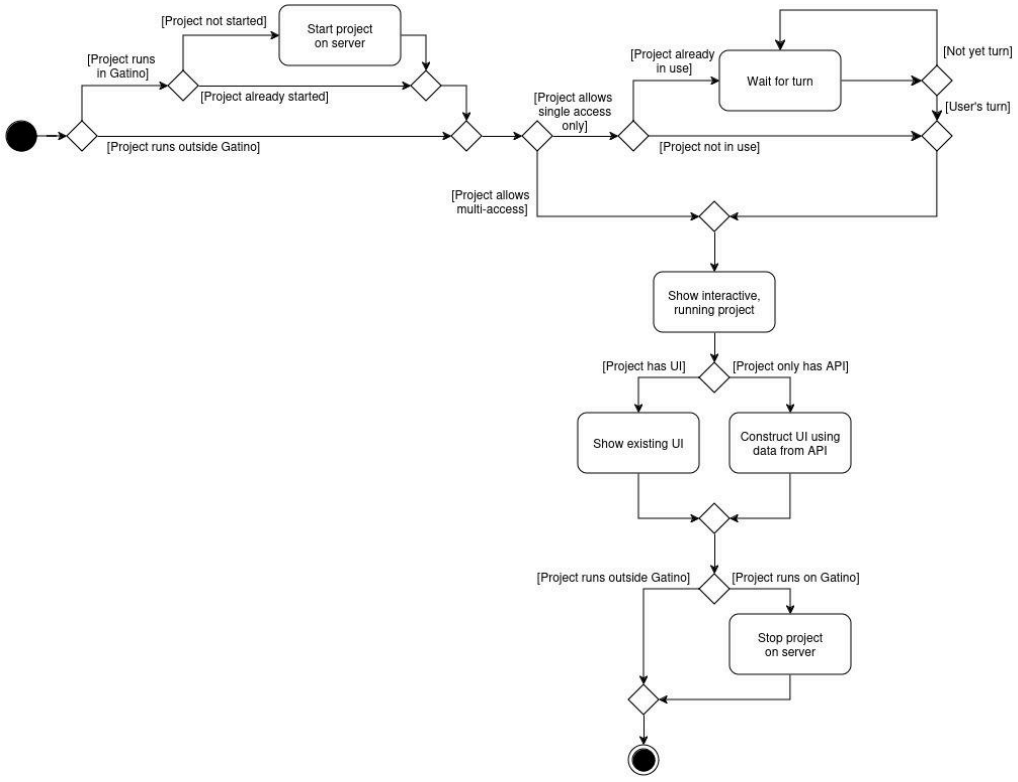

Fig 8: The workflow for task 2.3. - Run a demonstration of an IoT project

4 - Non-Functional (Quality) Requirements

Based on the ISO/IEC 9126 quality model (*ISO/IEC 9126-1*, n.d.), we present a structured presentation of the non-functional (quality) requirements, covering all of the defined characteristics and sub-characteristics:

Functionality

- Requirement**:** The system must provide complete functionality as specified in the requirements document.
    - **Suitability:** The search and demo features must be operational by the end of the development phase.
    - **Accuracy:** Search results must achieve at least 95% relevance as measured by user testing.
    - **Interoperability:** The system must successfully integrate with at least two external tools (e.g., authentication systems) by deployment.
    - **Security:** Implement role-based access control to ensure only authorized users can access sensitive features, validated through a security audit before launch.
    - **Functionality Compliance:** All functions must comply with specifications, verified by the Innovation Lab and product owner Dr Le Minh Duc.
- Verification: Conduct functional testing and gather feedback from stakeholders during the development process with a target completion date of two weeks before the final deployment.

Reliability

- Requirement: The system must maintain an uptime of 99.5% over any given month, with recovery from failures in under 10 minutes.
    - **Maturity:** All components must pass stability testing with no critical failures during the testing period.
    - **Fault Tolerance:** The system should handle up to 3 simultaneous failures without service interruption.
    - **Recoverability:** The system must restore operations within 10 minutes after any failure, validated through recovery drills conducted quarterly.
    - **Reliability Compliance:** Achieve at least 95% compliance with reliability metrics during testing.
- Verification: Monitor system uptime and conduct reliability testing frequently during the testing phase.

Usability

- Requirement: The user interface must achieve a usability score of at least 80% based on System Usability Scale (SUS) assessments from at least 30 users.
    - **Understandability:** At least 90% of users should complete key tasks without assistance during usability testing.
    - **Learnability:** New users should complete uploading their projects with an average time of less than 10 minutes and searching for less than 2 minutes.
    - **Operability:** Users must complete critical functions within 2 clicks on average.
    - **Attractiveness:** The interface must receive a satisfaction rating of at least 80% for visual appeal in user surveys.
- Verification**:** Conduct usability testing sessions within three weeks of the initial development phase, and collect client and user feedback.

Efficiency

- Requirement: The system must support 100 concurrent users, with response times of less than 5 seconds for search queries and demo requests.
    - **Time Behaviour:** At least 95% of interactions must respond within the specified time frame under load.
    - **Resource Utilization:** CPU and memory usage must not exceed 75% under peak load conditions.
    - **Efficiency Compliance:** Achieve compliance with efficiency metrics during load testing.
- Verification: Use load testing tools to simulate user interactions and gather performance data two weeks before deployment.

Maintainability

- Requirement: The system must allow for critical updates to be implemented within 1 hour and feature updates within 24 hours.
    - **Analyzability:** All of the code must follow the conventional coding standard and Gatino coding standard, facilitating easy understanding.
    - **Changeability:** Changes to features should require no more than 5 hours of developer time on average.
    - **Stability:** No more than 2 critical issues should arise from modifications in 30 days following an update.
    - **Testability:** At least 80% of the code must have automated tests in place to ensure quick validation of changes.

- Verification: Review the coding rules and conduct maintenance tests one month after deployment.

Portability

- Requirement: The system must be deployable across at least two different operating systems (e.g., Linux and Windows) with minimal configuration changes.
    - **Adaptability:** The system must function correctly in each environment with less than 5 configuration changes required.
    - **Installability:** The installation process should not exceed 30 minutes for both operating systems.
    - **Co-existence:** The system should operate without conflicts alongside existing applications in the target environments.
    - **Replaceability:** Components should be replaceable without requiring more than 1 hour of downtime.
- Verification: Conduct compatibility testing on both operating systems, documenting configuration requirements and installation times within three weeks of deployment.

5 - Interface Requirements

In this section, we will describe the communication interfaces within the software during operation.

5.1. - System in Context

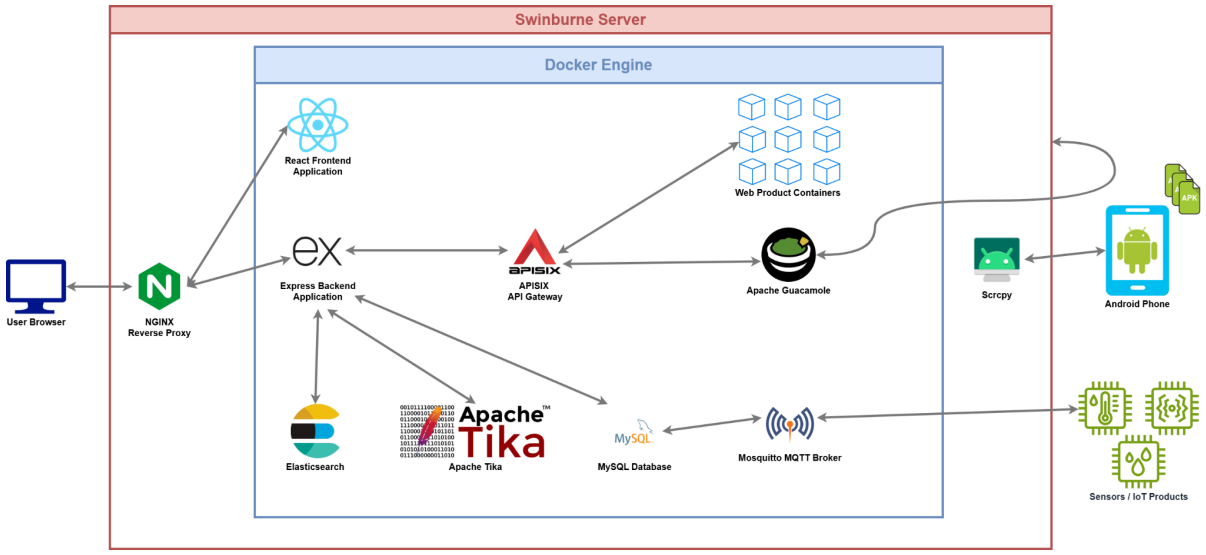

Fig. 1: The system diagram. The full-resolution diagram can be seen here.

The system diagram above works for the Search, Demo, Management features as follows:

1. **Search feature**:
    a. **Updating search information**: A student or system admin sends a request to add/update/delete a project through the Project Management Screen. The server's API gateway will redirect the request to the Project Management Module where it is validated and the Project database is updated. This module then invokes the Search Module to update the project's search information in Elasticsearch. The Search Module will use the Data Extraction Module to extract and refine the search information for the project before it sends it to Elasticsearch.
    b. **Search for projects**: A user sends a search request to the server through the Search Screen. The server's API gateway will redirect the request to the Search Module, which constructs the necessary query to search the projects in Elasticsearch. It then returns the resulting projects to the user.
2. **Demo feature**: A user requests a project demo through the Demo Screen. The server's API gateway will redirect the request to the Demo Module, which runs the corresponding project container. Before and after a demo is run, the Demo Module

may need to turn the container on or off as necessary to save resources while also maintaining a good user experience.

5.2. - User Interfaces

As Gatino is primarily a web-based platform, its user interface will be presented in the browser. We will use the Mantis theme (*Mantis React Admin Dashboard*, n.d.) chosen by the client to develop the user interface.

For the search feature, the user will be presented with a search bar where they can enter text search queries similar to Google or Bing search. In addition to the search bar, they can also use dropdowns and checkboxes to provide more specific search options that refine the results. After the user submits the search query, they will see the results presented in a compact and easy-to-navigate grid. The picture below shows an example of the Search screen, made using the Mantis theme.

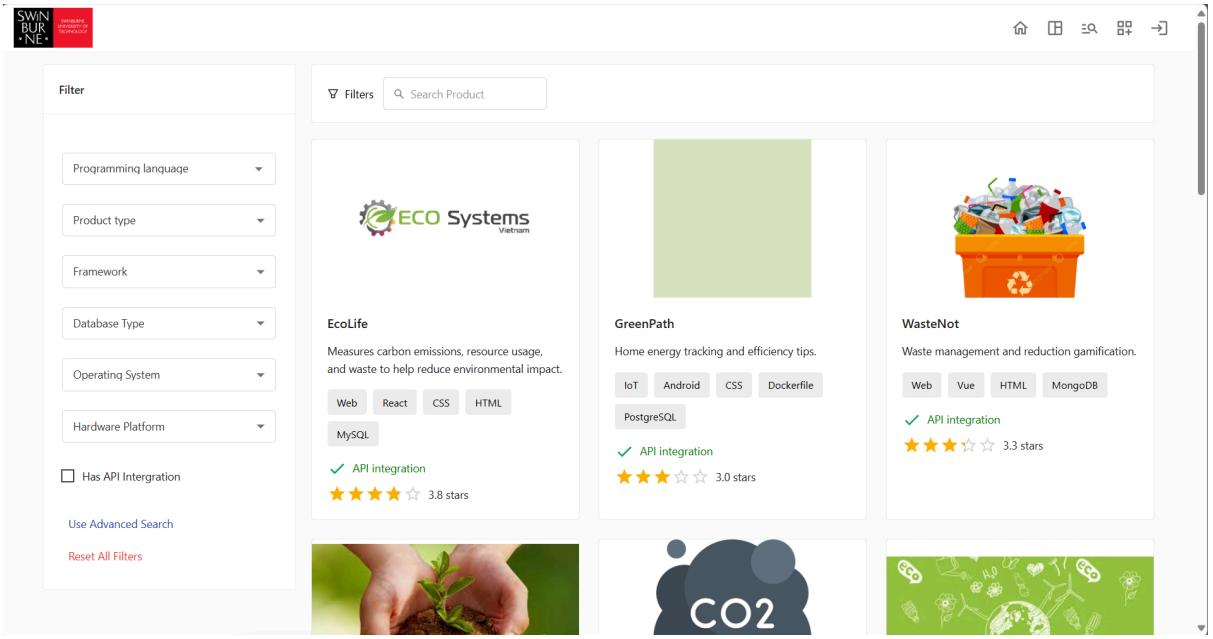

Fig. 2: The Search screen with the Mantis theme.

For the Demo feature, the user will first have to enter the Project Details page, where they will see a “Run Demo” button if the project supports it. After clicking on this button, the user will be directed to a webpage running a functional demonstration of the project. For projects that can only be demonstrated to one user at a time, the user will be instructed to wait for others to finish their sessions before they can start playing with the demonstration.

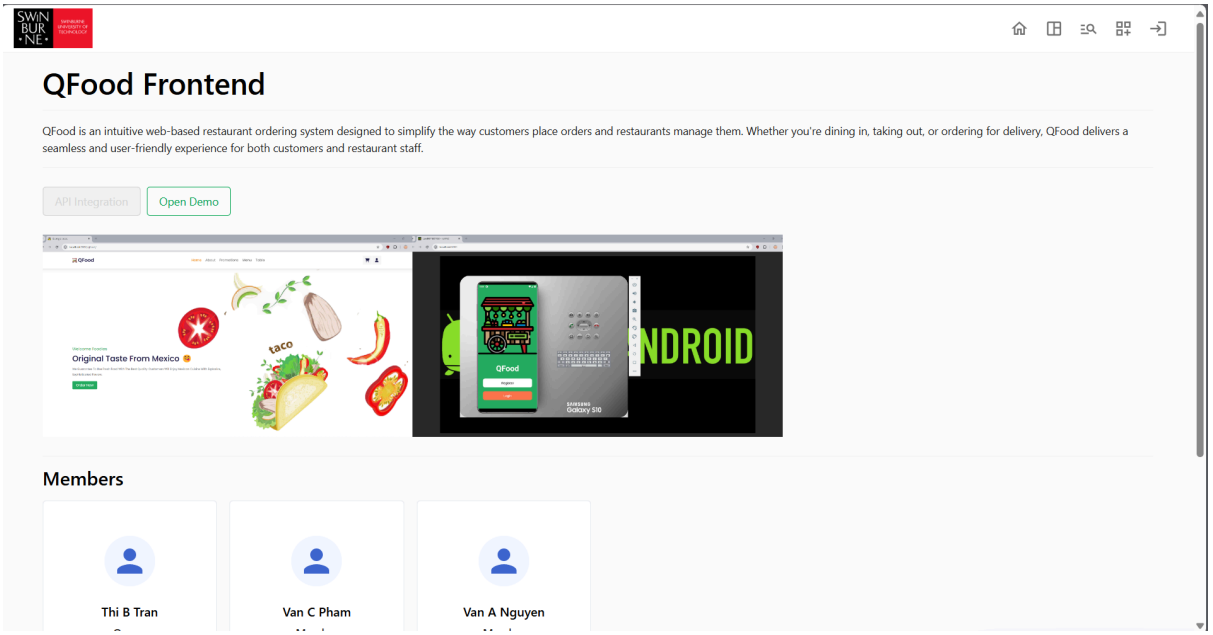

Fig. 3: The Project Details page with the Mantis theme showing a “Run Demo” button.

The UI of the actual demonstrations will depend on the type of the project. For web-based project demos, the UI will be that of the website the students built. For Android-based demos, the UI can show either an emulator or a mirrored real device running the app.

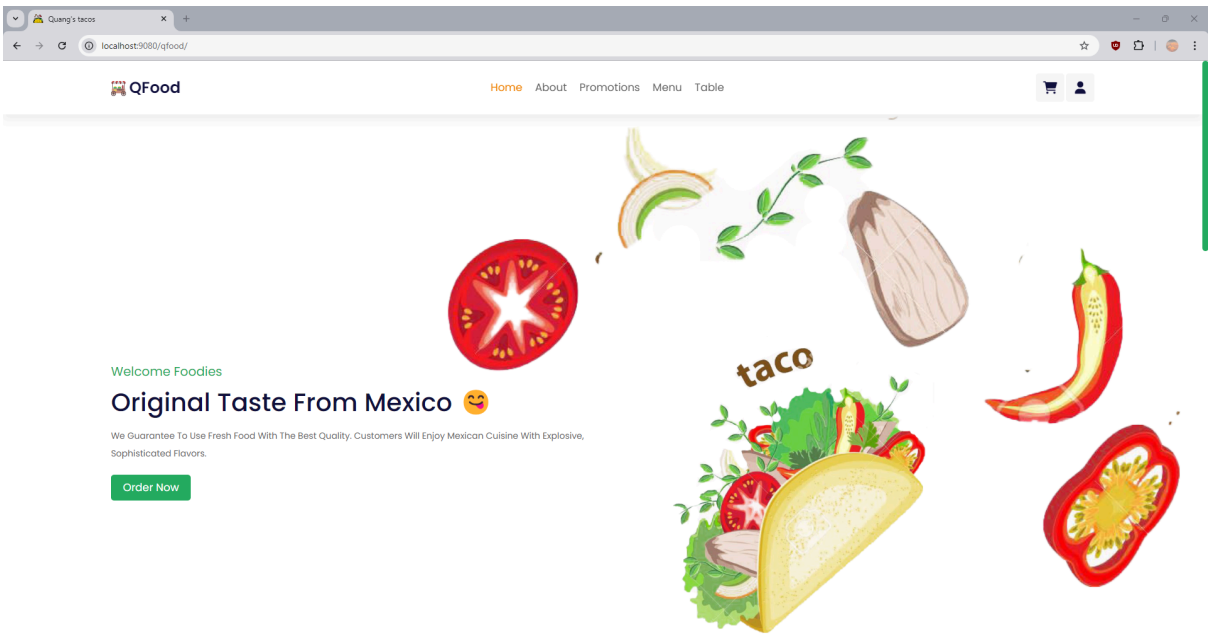

Fig. 4: A running demonstration of a web project.

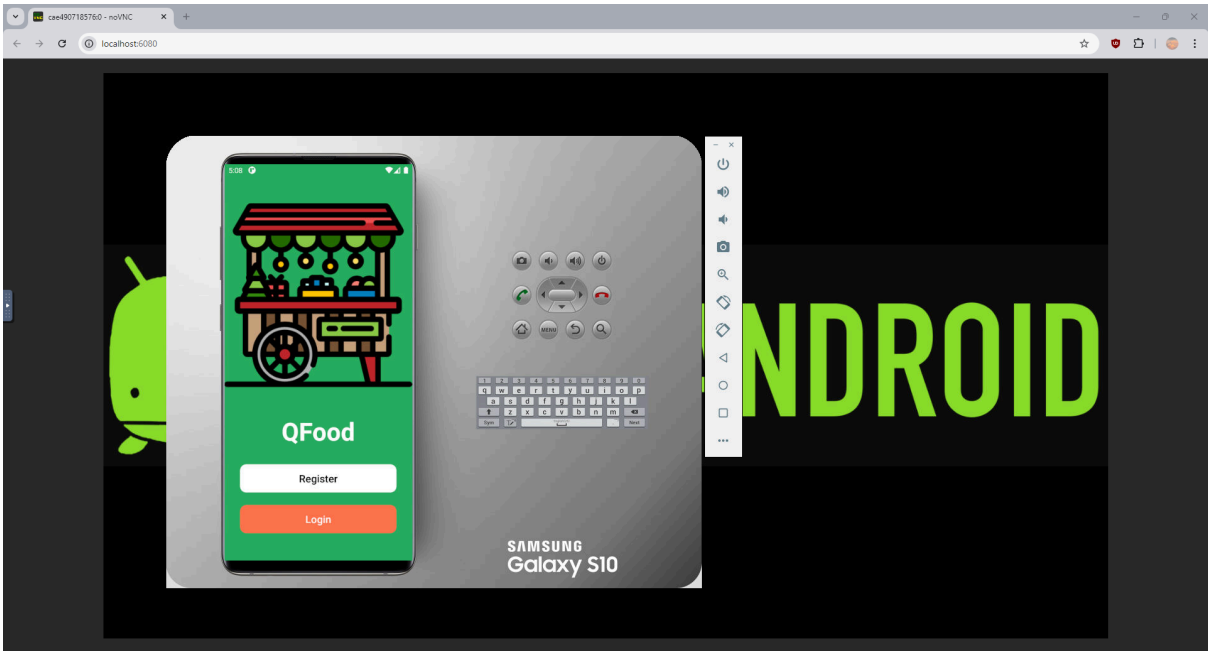

Fig. 5: A running demonstration of an Android project on an emulator.

5.3. - Hardware Interfaces

This software does not have specific hardware interface requirements. However, an Android phone is connected to the Swinburne server via USB. The server utilizes Scrcpy to mirror and display the phone's user interface..

5.4. - Software Interfaces

MySQL

- **Type**: Private, self-deployed as a Docker container.
- **Version**: 8.0.39
- **Source**: Official image from the Docker registry.
- **Purpose**: Store projects' information.
- **Communication mechanism**: Only Gatino's backend modules can interact with the MySQL database (for security reasons). Communication will be performed by a client library of the chosen backend programming language (such as MySQL2 or Sequelize for JavaScript/TypeScript). The database service will listen on port 3306.

Elasticsearch

- **Type**: Private, self-deployed as a Docker container.
- **Version**: 8.15.2
- **Source**: Official image from the Docker registry.

- **Purpose**: Store projects' search information and act as a search engine.
- **Communication mechanism**: Only the Search module of the backend can interact with the Elasticsearch service. Communication will be performed by a client library of the chosen backend programming language (such as @elastic/elasticsearch for JavaScript/TypeScript). The Elasticsearch service will listen on port 9200.

GitHub API

- **Type**: Public API owned by Microsoft.
- **Version**: Latest, 2022-11-28.
- **Source**: None.
- **Purpose**: Extract technical metadata of a project if the student has provided a public GitHub repository link.
- **Communication mechanism**: Only used by the Data Extraction module of the backend to get a project's technical metadata for use as search information. The backend module will send an HTTPS request to GitHub's RESTful API to obtain the necessary information.

Apache Tika

- **Type**: Private, self-deployed as a Docker container.
- **Version**: 2.9.2.1
- **Source**: Official image from the Docker registry.
- **Purpose**: Extract text content from student documents such as PDF, DOCX reports.
- **Communication mechanism**: Only used by the Data Extraction module of the backend to extract text from student documents for use as search information. The backend module will send an HTTP request to the Apache Tika service to obtain the necessary information. The service will listen on port 9998.

Docker Engine API

- **Type**: Private, part of the installed Docker Engine.
- **Version**: Dictated by the Docker Engine version on the Swinburne server.
- **Source**: None.
- **Purpose**: Turn on/off the containers that run the deployed innovation projects.
- **Communication mechanism**: Only used by the Demo module of the backend to turn on/off the Docker containers that host the innovation projects. Communication takes place over HTTP.

APISIX

- **Type**: Private, self-deployed as a Docker container.
- **Version**: Latest stable release.
- **Source**: Official APISIX image from the Docker registry.
- **Purpose**: API gateway for routing, load balancing, and authentication.
- **Communication mechanism**: Used by backend services to manage API traffic. Listens on ports **9080** (HTTP) and **9443** (HTTPS).

Guacamole

- **Type**: Private, self-deployed as a Docker container.
- **Version**: Latest stable release.
- **Source**: Official Apache Guacamole image from the Docker registry.
- **Purpose**: Remote desktop gateway for accessing project demos via web.
- **Communication mechanism**: Accessed via **HTTP/HTTPS** (port **8080** by default). Backend modules interact with Guacamole's API for session management.

Bitnami/etcd

- **Type**: Private, self-deployed as a Docker container.
- **Version**: Latest stable release.
- **Source**: Bitnami's official etcd image from the Docker registry.
- **Purpose**: Distributed key-value store for configuration management and service discovery.
- **Communication mechanism**: Used by **APISIX** or other services for dynamic configuration. Listens on ports **2379** (client) and **2380** (peer).

Mosquitto

- **Type**: Private, self-deployed as a Docker container.
- **Version**: Latest stable release.
- **Source**: Official Eclipse Mosquitto image from the Docker registry.
- **Purpose**: MQTT broker for lightweight IoT/pub-sub messaging.
- **Communication mechanism**: Used by backend or demo modules for real-time notifications. Listens on port **1883** (MQTT) and **8883** (MQTT over SSL).

5.5. - Communication Interfaces

The Gatino system utilizes several communication interfaces to facilitate secure data transmission and efficient system operation:

- **HTTP/HTTPS**: The platform communicates primarily through HTTP/HTTPS protocols, ensuring secure transmission of data between client browsers and the server. This setup enables all web-based interactions, including user queries and demo requests.

- **RESTful API**: The search and demo features are exposed through a RESTful API, accessible only to the Gatino system for security purposes. The API uses HTTPS to ensure secure data exchange between the Gatino platform and other internal components.
- **WebSocket**: For real-time updates, such as queue status and demo interactions, WebSocket connections are established to provide low-latency communication, enhancing user experience, particularly during live demo sessions.
- **Database Communication**: Communication between the Gatino system and the database (MySQL) leverages secure database protocols, with encryption applied to sensitive data transactions. This ensures the integrity of project data and the secure retrieval of indexed content.
- **GitHub API**: To facilitate the extraction of project metadata from GitHub repositories, the system uses the GitHub API. This API communication is secured via access tokens, ensuring that only authorized requests are processed, thus protecting project data integrity.
- **MQTT**: Used for lightweight, publish-subscribe messaging in scenarios requiring efficient IoT or device communication (e.g., monitoring system health, sensor data, or asynchronous event notifications). MQTT ensures minimal overhead while supporting QoS levels for reliable delivery.

6 - References

ISO/IEC 9126-1:2001. (n.d.). ISO. Retrieved October 10, 2024, from

https://www.iso.org/standard/22749.html

Lauesen, S. (2001). Tasks & Support Task Descriptions as Functional Requirements.

Mantis React Admin Dashboard. (n.d.). https://mantisdashboard.io/