EEET2582/ISYS3461

Software Requirements Specification

Version 1.1

DEVision - Job Manager Subsystem

Table of Contents

Introduction

Vietnamese tradition emphasizes the proverb **'Settled and Flourished'** (*An Cư Lạc Nghiệp*), highlighting the necessity of establishing a solid and fulfilling career as the foundation for prosperity and personal happiness. **DEVision** was created to embody this principle by **bridging job applicants** in the Computer Science fields with their potential employers. Our primary goal is to empower applicants to find roles that align with their technical skills and career aspirations, while simultaneously assisting companies in attracting top talents that match their specific technical demands and long-term vision. To achieve this dual mission, DEVision is architected into two core components: the **Job Applicant** subsystem and the **Job Manager** subsystem.

The **Job Applicant subsystem** is designed to provide comprehensive tools for career advancement. It handles core functions like registration, secure authentication, job search, and job application submission. Applicants benefit from the system's ability to help them find ideal roles quickly. By paying a **monthly subscription**, Applicants gain access to a powerful premium feature: **real-time notifications** of new job posts. This feature alerts applicants when a job appears that matches their predefined criteria, including desired technical background, employment status, salary range, location, and job titles.

The **Job Manager subsystem** provides a dedicated platform for companies to manage their recruitment needs efficiently. This subsystem facilitates company registration, secure authentication, detailed company profile creation, job post creation, application processing, and the ability to proactively search the applicant database. Companies can also benefit from a **premium monthly subscription**. This feature provides **real-time notifications** when new job applicants register or update their profiles, matching the company's specific headhunting criteria for technical skills, desired salary, location, and educational background. This feature helps companies to identify talents as soon as they appear.

This document details the required specifications for the Job Applicants subsystem, covering Architecture, Data Model, Functional, Project Management, and Contribution aspects. To facilitate staged development and complexity scaling, the Architecture and Functional requirements are categorized into three difficulty levels: **Simplex** (Basic), **Medium**, and **Ultimo** (Advanced).

You are highly recommended to analyze the requirements of all three difficulties to decide the target complexity, capability, flexibility, and goals of your system architecture and application.

Contribution Requirements

Every team member is expected to contribute to ALL the following tasks by producing recognizable deliverables or providing substantial feedback:

1. Design Data Model
2. Design System Architecture
3. UX/UI Design of ALL Requirements
4. Technical Report
5. Specialized in Front-end OR Back-end OR Full Stack Development

For development tasks, each member is expected to contribute either to individual or pair programming. The amount of fair contribution is determined by the complexity of the supported features and average workload of the entire team. **Final contribution assessment and weight distribution of grades for group assignments will be made at the discretion of the course coordinator**

If there has been a problem with group members not completing their allocated tasks, inform your Course Coordinator as soon as possible so staff can intervene before the problem escalates.

Software Development Methodology Requirements

This section specifies the compulsory methodological framework and teamwork standards required for the execution and delivery of the DEVision project.

Collaborative Development: All teams are required to adopt an **iterative development** approach, such as SCRUM or Kanban board, to enable periodic collaboration and advance the project requirements incrementally.

To ensure transparency and traceability, every team must utilize a **digital project management tool** (e.g., Jira, Trello, GitHub Board) to rigorously track all project tasks. This board must document individual tasks, track team commitment, progress, and verify the completion status of all work items.

Each development Team or Squad must include the **lecturer** as a participant on their digital project board to enable real-time monitoring and support.

Communication and Review Cycle: A formal, recurring review cycle is mandatory to maintain alignment and quality. **Every two weeks**, each team is required to schedule and conduct a combined **Sprint Review and Retrospective Meetings** with the lecturer. These sessions serve as critical milestones where the team must formally present recent progress, discuss lessons learned and experiences gained, and proactively identify and raise any significant project impediments or risks.

Professionalism: Teamwork performance will be comprehensively evaluated based on three core dimensions: **Contribution to Development** (code quality and feature implementation), **Management Adherence** (quality of board updates and task tracking), and **Involvement** (active participation in all meetings and discussions).

Meeting Attendance Policy: **Punctuality and presence at scheduled review meetings are paramount.** Should any team member be unable to attend a meeting with the lecturer, they are required to provide notification **as soon as possible or at least 24 hours in advance** with a **valid reason** and **supporting evidence**.

Any absence **without prior notification AND valid justification** will automatically result in **a substantial penalty** applied to the individual's teamwork assessment criteria.

Milestone 1 Requirements

Due Date: 18:00 on 28th November 2025

In this section, every team is required to document the design decisions and justifications on the **Data Model** and **System Architecture** in a **30-page report**. The page count does NOT include your cover page and table of content.

You should analyze which of the following technologies are suitable for the application and design your architecture accordingly to leverage the potential of the chosen tech stack.

a/ **Frontend**

- React-based framework: e.g., React, Vite, Next.JS
- Vanilla (Plain) HTML, CSS, and Javascript

b/ **Backend**

- MEN Stack: MongoDB, ExpressJS, NodeJS
- Spring Boot
- Redis and Kafka

c/ **Deployment**: Docker or Kubernetes (Bonus)

R01: Design Conceptual Data Model for Database

Develop an Entity Relationship Model (ER Model) to store information necessary for fulfilling the requirements. Include this ER model in your project report.

- **Conceptual Modeling**: Design a high-level representation of your database schema using appropriate concepts and entities relevant to your system's requirements.

- **Data Requirements**: Ensure the model accommodates all necessary data elements and relationships essential for system functionality.

- **Documentation**: Briefly explain the design ideas and purposes of entities and relationships depicted in your conceptual data model. Justify the advantages and potential drawbacks of your data model design **in ONE designated section. Describing the advantages and drawbacks in different sections results in 2 points penalty for your Data Model Justification.**

R02: Present Architecture Diagram of the Entire System

You are required to separate the front-end and back-end systems in your deliverable. That is, the front-end system is not inside the back-end, and vice-versa.

Create an architecture diagram illustrating the entire system using either UML Component Diagram (Option 1) or C4 Model System, Container, and Component Diagrams (Option 2).

Option 1: UML Component Diagram

Use UML Component Diagram to depict the architecture of your system.

System Level:

- **Architecture**: Illustrate the sub-systems (front-end, back-end, databases, etc.,) in your DEVision application, plus external systems that your front-end and back-end services use.

Back-end:

- **Architecture**: Illustrate how components are organized and interact to deliver the services.
- **Utilities Classes and Configuration**: Show helper classes and configurations used in your backend (e.g., Security, Token Generator, Cookie Configurer).

Front-end:

- **Architecture**: Illustrate components, sub-components, and helper classes used in your web application.
- **Interaction with Back-end**: Express how front-end components communicate with the back-end (e.g., REST API calls), which backend components are the intermediate or final recipient of each front-end request.

Note:

- Depict the System Level in one diagram. Only specify the sub-systems, not their modules, in your diagram.
- Depict the back-end and front-end architectures in two diagrams.
- Only show the element of a higher-level hierarchy on the other subsystem that directly interacts with the currently focused subsystem. For example, if you are drawing the front-end architecture and express that the *User Authentication Service* in the front-end communicates with the *AuthenticationModule* of the back-end, show only the *FilterModule* and *AuthenticationModule* in the back-end subsystem as the recipient of the request.
- **Component Roles**: Describe the role and responsibilities of key components in the back-end and front-end of your system. Note that database is considered a back-end element.

Option 2: C4 Model Component Diagram

Use C4 Model Container and Component Diagrams to depict the architecture of your entire system.

Components Coverage:

- **Back-end and Front-end Components**: Represent components as described in Option 1.
- **Component Descriptions**: Include brief descriptions within each component regarding their purpose and functionality.
- Include sub-components in your C4 Component Diagram

General Requirements for Both Options:

- **Clarity and Conciseness**: Ensure the diagrams are clear, concise, and effectively communicate the system architecture. Your diagrams altogether shall explicitly reveal the **Layer-Based**, **Modular Monolith**, or **Microservice** design of your back-end, or the **Layer-Based**, **Componentized**, or **Headless UI** design of your front-end.
- **Alignment with Requirements**: Demonstrate how your architecture meets the specified system requirements and design principles.
- **Architectural Rationale**: Concisely justify the architectural choices made for each component. Justify the advantages and potential drawbacks of your data model design **in ONE designated section**. **Describing the advantages and drawbacks in different sections results in 2 points penalty for your Data Model Justification.**

  Your design justification shall consider the following six aspects:

  1/ **Maintainability**: Enables the system to be easily understood, updated, and tested independently by current and future Development Teams.

  2/ **Extensibility**: Enables new features to be added with minimal impact on existing components. This approach allows for future enhancements and adaptations to changing user needs or market demands.

  3/ **Resilience**: The system's ability to continue functioning correctly in the face of unexpected conditions, such as component failures or high loads.

  4/ **Scalability**: The capacity of a system to handle increased load or user demand without sacrificing performance

  5/ **Security**: The measures taken to protect a system and its users from unauthorized access, data breaches, and other threats.

6/ **Performance**: Performance relates to how quickly and efficiently a system responds to user requests and processes data.

These tasks aim to evaluate your ability to design a robust data model and articulate a coherent system architecture, which are essential skills in software engineering.

R03: Frequent Usage of GitHub

Your team must use GitHub as the only tool to store your deliverables, including diagrams, reports, database, and code base. You are required to use GitHub in a professional, iterative, and active approach by frequently pushing your primitive, partially completed, or finalized deliverables into the repositories, and use appropriate communication language in your repository.

Additional Note:

1) NOT Using GitHub frequently during the project results in up to 10 points penalty of your assessment grade.
2) In your report, submit the proof of GitHub contribution throughout the project duration
3) In Canvas, submit the zip file of your GitHub repository.

Missing either 2) or 3) **results in three points penalty each**.

Milestone 1 Presentation Requirements

Your Squad is required to present your **system architecture**, **data model**, and how the **frontends and backends architectures of the Job Manager and Job Applicant subsystems** provide the job posting, job searching, job application, and application approval features. Your Squad is also required to justify the **advantages** and **limitations** of your system design, which you can optionally use slides to organize your explanations.

After the presentation, each member will be questioned about the design decisions and justifications of your Squad's or subsystem's architecture. **Members who cannot explain the sections which they contributed to the report, system design and system analysis will be penalized**.

Milestone 2 Requirements

Due Date: 18:00 on 13 January 2026

1/ You are required to implement the architecture and features using the following technologies:

a/ **Frontend**

- React-based framework: e.g., React, Vite, Next.JS
- Vanilla (Plain) HTML, CSS, and Javascript

b/ **Backend**

- Database: MySQL, MongoDB, Postgres
- Spring Boot, ExpressJS, NodeJS
- Redis
- Kafka

c/ **Deployment**: Docker or Kubernetes

2/ Every Squad is required to include a database script to add the following initial data into the system:

a/ One ADMINISTRATOR who manages the system

b/ Four COMPANY accounts:

- Two **Freemiums**, each has at least two job posts with different job titles.
- Two **Premiums**, each has at least two job posts. Each company belongs to a different Computer Science domain, e.g., one Premium hires **Software Engineers**; the other hires **Data Engineers**.
- One **Premium** subscribes for talents knowing *React*, *Spring Boot*, and *Docker,* who want to work as a *Full-time Intern* **Software Engineer** in *Vietnam* with a desired salary of more than 800 USD.
- One **Premium** subscribes for talents knowing *Python*, *AWS*, and *Snowflake,* who want to work as a *Contractual* **Data Engineer** in *Singapore* with a desired salary of more than 1200 USD.

c/ Five APPLICANT accounts:

- Two are from Vietnam, two are in Singapore, one is from another nation.
- Two Premium accounts, one from Vietnam, the other from Singapore

Architecture Requirements.

You are strongly recommended to read through requirements of ALL three Levels and the rubric to decide your project goals and plans.

Role Specification

- APPLICANTS: Individuals who apply for jobs
- COMPANIES: Organizations posting job positions and filtering applications of their job posts.
- ADMIN: Individuals managing applicants, job posts, and companies.

Level	Feature ID	Description
Simplex	**A.1.1**	Apply N-Tier (Layer-based) Architecture to divide your front-end AND back-end into different layers. Each layer has a designated concern and stores all classes addressing that concern across different business features.
	A.1.2	A typical Layer Hierarchy is **Presentation**, **Business Logic**, and **Data Access**. However, to reuse query operations in the back-end and separate query execution from Data Access Object (DAO), **you are required to implement a Repository Layer**, where queries statements are defined and executed.
	A.1.3	Any Feature that does not follow the Tier structure shall be organized separately. E.g., Cookie Configuration, Filter.
Medium Backend	**A.2.1**	Apply Modular Monolith Architecture to your back-end. Each module has a bounded context to handle a specific (group of) business feature. A module stores all classes of the N-Tier Layers or another design pattern to provide the business feature.
	A.2.2	Within a Module, the access rules shall go from one layer to another. Specifically, the **Presentation** Layer shall communicate with the **Business Logic** Layer which calls **Repository** method to execute queries. The **Presentation** Layer must NOT be able to access the functions provided by the **Repository** Layer.
	A.2.3	Each Module contains two groups of **Service** APIs: 1. **External APIs** to expose public services for other modules in the system to invoke 2. **Internal APIs** to be used by the Presentation Layer or other Services of the module. Internal API services are NOT accessible to other modules

	A.2.4	Each module exposes its internal and external API **<u>via interfaces</u>**. The Presentation Layer or other modules shall NOT call the Service Layer of the module to invoke the APIs' services.
	A.2.5	The backend must present only necessary data when responding to a request from the front-end. For example, when showing the APPLICANTS matching the search criteria, their credit card number shall not be exposed in the response.
	A.2.6	You must also organize your DTOs into internal and external DTOs. External DTOs are accessible by other modules, while internal DTOs are only accessible by classes within the module.
Medium Frontend	**A.2.a**	In the frontend, you should define common display elements (e.g., Buttons, Labels) as configurable components in a separate folder, so they can be reused by different pages without redefining the style again.
	A.2.b	Componentize your Front-end, such that each component has a package containing the following elements: 1. Component's presentation in HTML or JSX 2. Sub-components 3. Back-end service calls 4. Event handlers on the component 5. CSS styling, if applicable All the five elements are NOT placed in the same Component JavaScript file. These five elements shall be placed in separate files under the same package.
	A.2.c	The front-end must contain a REST HTTP Helper Class to provide a base function for conducting REST Requests such as GET, POST, PUT, PATCH, and DELETE, along with parameters of request headers and request body.

Ultimo Backend	**A.3.1**	Apply Microservice Architecture to your back-end. Each service has a bounded context to handle a specific (group of) business capability. To achieve this, a microservice typically stores all classes of the N-Tier Layers for a single or group of related business capabilities, stores a module of a business capability from the modular monolith architecture, or applies a different design pattern to provide business capabilities.
	A.3.2	Communication among microservices shall be conducted via a Message Broker, e.g., Kafka.
	A.3.3	Every microservice manages its own database(s). Each microservice contains a specific connection String to its database.
	A.3.4	Your team must apply database sharding techniques to partition and query different shards. This requirement aims to test your design in terms of scalability and efficiency.
	A.3.5	**(Bonus Feature)** Your database must execute transactions instead of queries, hence the database can rollback transactions when an unexpected error occurs.
Ultimo Frontend	**A.3.a**	**Apply Headless UI to your frontend architecture** for every UI elements that share similar templates or actions, but can be customizable to suit with the context of the page. For example, the Table to show job posts and the Table to show applications to a job post can share the same data template: heading is the title of the job or the name of the applicant; A View Button to see the content of the job post or application, and a Delete Button to remove the job post or application from the current list. In this case, the Table can be a Headless component, where the Job Post List Page and Application Page use the Table to render the Job Post and Applications accordingly.

Job Manager Functional Requirements

1/ Company Registration

Level	Feature ID	Description
Simplex	**1.1.1**	The system shall provide a registration form requiring Email, Password, and Country as mandatory fields. Phone number, Street name/number, and City are optional fields.
	1.1.2	The system shall enforce a unique email constraint, ensuring no two companies share the same email address in the database.
	1.1.3	The system shall send an email to the new Company with information on how to activate the account. A Company can only login after they activate their account.
	1.24	The Country input field must be a selectable option (e.g., a dropdown list) instead of a direct text input field.
Medium	**1.2.1**	The system shall validate password strength against the following criteria: a) At least 8 characters. b) At least 1 number. c) At least 1 special character (e.g., $#@!). d) At least 1 capitalized letter.
	1.2.2	The system shall validate the email syntax to ensure it meets standard formatting requirements, including a) Contains exactly one '@' symbol. b) Contains at least one '.' (dot) after the '@' symbol. c) The total length of the email address must be less than 255 characters. d) No spaces or prohibited characters (e.g., ( ) ; :).
	1.2.3	The system shall validate the Phone number (if provided) to ensure it: a) Contains only digits and must start with a valid international dial code (e.g., +84, +49). b) The length of the digits following the dial code must be less than 13.
	1.2.4	All input validations specified above must be performed on both the frontend for quick user feedback and the backend for security. Any validation error must be clearly displayed to the end-user.
Ultimo	**1.3.1**	The system shall support registration via SSO (Single Sign-On) from ONE selected external platform: Google, Microsoft, Facebook, or GitHub. You can use personal email in this project.
	1.3.2	Upon successful SSO registration, the system shall persist the core company profile (name, email, country) in the server. The system must be

		configured so the company that logs in using the SSO interface cannot use a password for Job Manager system access.
	1.3.3	The system shall select a User's attribute as the sharding key to partition and store the user data in the designated database shard. The chosen shard key must enhance the search algorithm of the Job Seeker and/or Job Manager systems

2/ Company Login

Level	Feature ID	Description
Simplex	**2.1.1**	The system shall authenticate the company using the submitted email and password. If the system does not use HTTPS, the frontend shall send credentials using the Basic Authentication format. If HTTPS is applied, the email and password can be sent in plaintext within the request body.
	2.1.2	Upon successful login, the backend shall generate a JSON Web Signature (JWS) token. This token must contain signed user identity data (e.g., user ID, Role) to verify its integrity.
Medium	**2.2.1**	Upon successful login, the backend shall generate a JSON Web Encryption (JWE) token instead of a JWS. This token must contain user identity information (e.g., user ID, Role) and ensure that the payload is encrypted and cannot be read by unauthorized parties.
	2.2.2	The system shall implement a mechanism to prevent brute-force attacks on the login endpoint, blocking the authentication for that account after 5 failed login attempts within 60 seconds.
	2.2.3	The system shall invalidate and revoke the user's JWE token when the user explicitly logs out or after the defined expiration time.
Ultimo	**2.3.1**	The system shall support logging in using an account from the single selected external platform defined in requirement 1.3.1 (Google, Microsoft, Facebook, or GitHub).
	2.3.2	The system shall use a dedicated Redis cache instance to store and quickly check the revocation status of the JWE tokens for non-SSO accounts, improving security performance.
	2.3.3	The backend shall implement a token refreshing mechanism for non-SSO accounts. The system issues a short-lived Access Token and a longer-lived

		Refresh Token to maintain user session without requiring frequent re-authentication.

3/ Profile Management

Level	Feature ID	Description
Simplex	**3.1.1**	Companies shall be able to **edit their core contact information**: Email, Name, Password, Phone Number, Address, City, and **Country**.
	3.1.2	The system must allow the companies to create and configure their **public profile**, including Company Name, About Us, and 'Who we are looking for' section to describe the desired personalities of potential employees.
Medium	**3.2.1**	The Company shall be able to **upload a Logo picture**. The system must automatically resize the image to a defined standard size for optimal display performance.
	3.2.2	The company shall be able to **upload images and videos** to showcase their events, mottos, or activities.
Ultimo	**3.3.1**	Any profile modification must be **persisted immediately** to the appropriate database shard. **Specifically**, if the job seeker changes the **Country** field, the application logic must perform a **data migration** of the entire user record to the new, corresponding database shard.

4/ Job Post Management

Level	Feature ID	Description
Simplex	**4.1.1**	The system shall allow the company to manage **<u>public</u>** and **<u>private</u>** job posts. Each job post has *Title, Description, Employment Type, Posted Date, Expiry Date* (optional), *Salary*, and *Location*. A job post is not publicly visible if it is not published by the company. Salary must be a **Range** (e.g., 1000 – 1500 USD), **Estimation** (About 1000 USD, Up to 2000 USD, From 3000 USD), or **Negotiable**. *Employment Type* includes **Full-time/Part-time; Internship, and Contract**. A job post can be Internship and Contractual at the same time, but it can only be Full-time or Part-time.
Medium	**4.2.1**	The system shall allow the user to **tag and manage a list of Technical Skills and Competencies for each job post** (e.g., adding tags like "Python", "Kafka", "SQL").
	4.2.2	The system shall display the applications of every job post, including the **Cover Letter** if applicable. The applications are organized into ***Pending*** or ***Archived*** categories. ***Pending*** are submitted applications to the job post. The newest application is placed on top of the ***Pending*** list. The Company is allowed to archive the application, which places the selected application into the ***Archived*** list. **The Job Applicant team must be responsible for persisting the data of each job application**
Ultimo	**4.3.1**	Profile updates must be propagated to a **Kafka topic** when the company changes skills or country of a job post. This enables all subscribed **Job Applicants** to be notified instantly if the applicant's technical background and country match the job post criteria.
	4.3.2	The system shall enable the user to **see the Curriculum Vitae (CV) and Cover Letter <u>files</u>** of each application.

5/ Applicants Search

Level	Feature ID	Description
Simplex	5.1.1	Companies shall be able to search for Job Applicants using four criteria: *Location (*City or Country*)*, *Education* (Bachelor, Master, Doctorate), *Work Experience* (None, Any, or contain specific keywords). The system shall allow the user to select multiple Employment Types in a single search request
	5.1.2	The Work Experience search must be **case-insensitive** (e.g., "Software Engineer" and "software engineer" yield the same results).
	5.1.3	**In any search request**, the user only provides one value for the **Location** (either one City or one Country).
	5.1.4	The search result shall display the **First Name**, **Last Name**, **Email**, **City**, **Country,** and the **Highest Education Degree**. Clicking on the Applicant shows the above information along with their **Education**, **Work Experience,** and **Objective Summary**
Medium	5.2.1	The system shall implement a Full-Text Search (FTS) capability on the Applicant data, **allowing searches across the Work Experience, Objective Summary, and Technical Skills fields**.
	5.2.2	The system shall allow job seekers to filter jobs based on **Technical Skills tags.** For example, if the user searches for *Kafka*, *Java*, and *MongoDB* tags, any applicant declaring their skills on *Kafka* **OR** *Java* **OR** *MongoDB* are included in the result.
	5.2.3	The frontend shall **lazy load** Applicants matching the search result criteria
	5.2.4	The page that provides Applicants search and result display shall be **responsive to desktop** and **mobile devices**.
	5.2.5	The search results shall display the **Skill Tags** (e.g., React, Kafka) of each applicant item.
Ultimo	5.3.1	The search query must be optimized to perform retrieval using a database sharding key (Country), ensuring searches are routed only to the relevant data shard. The system sets the default location of an applicant search to Vietnam.

	5.3.2	The system allows the user to mark an Applicant as *Warning* or *Favorite* when viewing the Profile or Job Application of an Applicant. The *Warning* or *Favorite* status of an applicant must be indicated in the result.

6/ Premium Company Subscription

Level	Feature ID	Description
Simplex	**6.1.1**	The system shall provide a **monthly subscription feature** where company can pay a fee of **30 USD** using a **third-party payment system** (e.g., Stripe, Paypal). The system must **record and display the premium status** on the company Profile page.
	6.1.2	The system must notify the company via **email** when their subscription is 7 days from expiration and when it has officially expired.
Medium	**6.2.1**	The system shall allow companies to define and save an **Applicant Searching Profile** including desired **Technical Background** (See 6.2.2), **Employment Status** (See 6.2.3), **Country**, **Salary Range** (See 6.2.4), and **Highest Education Degree**
	6.2.2	The system shall record desired **Technical Background as tags** (e.g., Kafka, React, Spring Boot) for efficient matching.
	6.2.3	The system shall record desired **Employment Status** as **multiple selections** from *Full-time*, *Part-time*, *Fresher*, *Internship*, and *Contract*. If **neither Full-time nor Part-time** is specified, the matching logic must include both Full-time and Part-time jobs.
	6.2.4	The system shall record acceptable **salary range** from a **minimum** to a **maximum** amount. If the minimum is not set, the minimum is 0. If maximum salary is not set, there is no upper limit. The search must also include applicants with **undeclared preferred salary**.
Ultimo	**6.3.1**	The system shall implement a real-time notification service using the Kafka messaging platform. A dedicated Kafka consumer service must instantly evaluate all new or edited applicant profiles against the criteria of all active

		premium companies. For every found match, the system must deliver an immediate real-time notification to the companies.

7/ Payment Service Development

Level	Feature ID	Description
Simplex	**6.1.1**	The Job Manager team shall develop the Payment API that allows **Applicants** in the *Job Applicant* system and **Companies** in the *Job Manager* system to subscribe for the real-time desired job notification service. The Payment API leverages the **third-party credit-card payment system stated on 6.1.1.**
Medium	**6.1.2**	The system must record the payment transaction with the applicant/company email, transaction time, and transaction status.

API Provision Requirements

Team Job Manager **must provide the following Service APIs to Team Job Applicant**:

1. Company Public Profile Data
2. Job Post Data
3. Subscription Payment API
4. Job Manager System Authorization (whenever applicable)

API Usage Requirements

Team Job Manager must NOT develop the following APIs. Instead, **team Job Manager must call the following Service APIs from Team Job Applicant:**

1. Applicant Profile Data
2. Job Application API, i.e., who applied for which job post of a Company
3. Job Applicant System Authorization (whenever applicable)

Deployment Requirements

Level	Feature ID	Description
Simplex	**D.1.1**	The frontend and backend systems are deployed in development environment.
	D.1.2	The Job Applicant system communicates with the Job Manager to provide and consume APIs for providing the software features (See API Provision and API Consumption Requirements)

Medium	**D.2.1**	The frontend, backend, database, redis cache, Kafka, API Gateway, Service Discovery, and other involved subsystems **are Dockerized.** **The components of the subsystem is hosted in one or at most 2 different physical or virtual machines.** For example, the backend, frontend, and redis are hosted in one computer, while Kafka, API Gateway, and Service Discovery are hosted in another computer.
Ultimo	**D.3.1**	Communication between Job Applicant and Job Manager, or Communication among Microservices applies API Gateway and Service Discovery.
	D.3.2	Kafka is applied for publish/subscribe feature and communication between the Job Applicant and Job Manager system.
	D.3.3	To simplify the setup, **API Gateway and Service Discovery can operate together in one physical machine**, but they are **hosted separately** from the frontends and backends. **Kafka is hosted separately** from the backends, frontends, API Gateway, and Service Discovery. In each subsystem, **backend microservices** must be hosted separately in **at least two** machines or cloud infrastructures.
	D.3.4	**Bonus Feature**: Deploy the DEVision ecosystem using Kubernetes

Milestone 2 Presentation Requirements

Your Squad is required to deploy and demo your **DEVision** system with the integration of **Job Manager and Job Applicant subsystems** for providing features listed in the functional requirements.

After the presentation, each member will be questioned about their contribution to the development, API Provision and API Usage. **Members who cannot explain or respond to the lecturer's questions will be penalized** by deducting the corresponding grade on the frontend, backend, or architecture.

Penalty

P1) NOT Using GitHub frequently during the project results in up to **10 points penalty** of your assessment grade.

P2) In your Contribution Declaration Sheet, submit the proof of GitHub contribution. You will need to **set your repository to public** to collect the contribution data. State the name of every member if the GitHub username does not contain your first name AND last name. **Only publish your repository at most 24 hours before submission. Missing the GitHub contribution results in 3 points penalty.**

P3) NOT submitting the **zip file** of your Squad's final deliverable on GitHub results in **3 points penalty**

P4) NOT submitting the **Project Charter** before Milestone 1 results in **3 points penalty**

End of Requirement Specifications