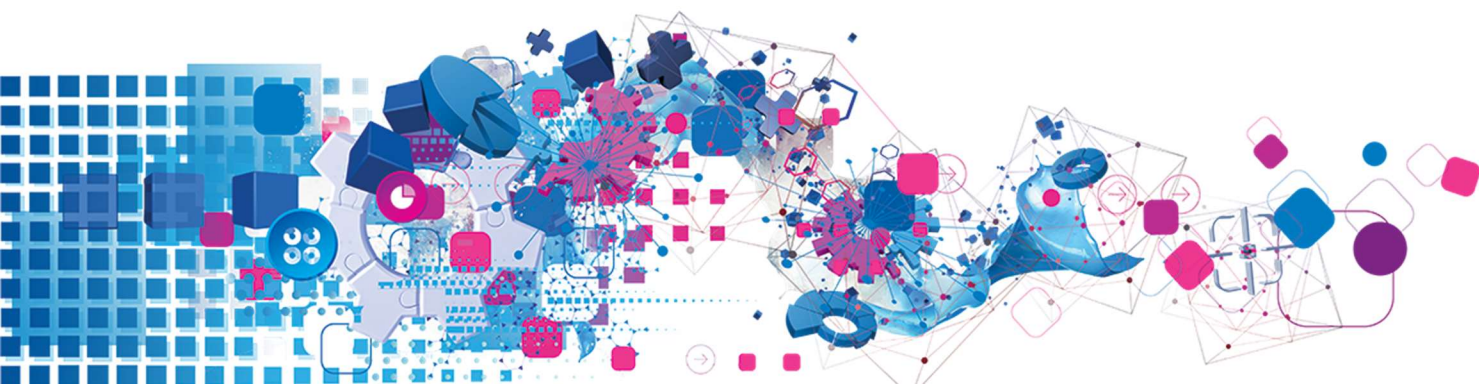




# Technical Assessment

Intermediate Java Developer

---



# Contents

- 1. Scope ..... 3
  - 1.1 Back-end ..... 3
  - 1.2 API ..... 3
  - 1.3 Front-end UI ..... 3
- 2. Business Requirements ..... 3
- 3. Functional Requirements ..... 4
- 4. Non-Functional Requirements ..... 5
- 5. Implementation ..... 5
- 6. Bonus ..... 5
- 7. Additional Notes ..... 5
- 8. Appendix A - Wireframes ..... 6
  - 8.1 Search Student ..... 6
  - 8.2 Create Student ..... 7

# 1. Scope

We require a distributed system to capture student details as well as their associated scores. Every student has an “active” current score along with any previous scores being archived within a data store for further processing.

The system will consist out of 3 parts namely Back-end, API Integrations and the Web Front-end. These 3 parts, should comply to the following pre-requisites:

## 1.1 Back-end

- Java 8 or higher needs to be used.
- Separation of Business and Data Access layers
- Postgres Database (or similar technology)
- Dockerized environment

## 1.2 API

- RESTful API (JSON) linked to Back-end
- APIs locally hittable using webservice clients such as Postman

## 1.3 Front-end UI

- Vue.js JavaScript Framework (or similar technology)
- Should use APIs to interact with Back-end
- Dockerized environment
- See sample wireframes in: [Section 8 - Appendix A - Wireframes](#)

# 2. Business Requirements

The system needs to provide the user with the following functionality:

- Register a new student profile
- Update student profile
- Delete student profile
- Capture a specific student's latest score
- View student profile including current and average scores
- Search student profiles registered on the system

### 3. Functional Requirements

- Student profile fields
  - Student Number (system calculated)
  - First Name (alpha characters only)
  - Last Name (alpha characters only)
  - Date of Birth (DD/MM/YYYY formatted date)
  - Cellphone Number (numeric characters including plus ( + ) sign allowed)
  - Email Address (alphanumeric including selected special characters applicable to email addresses)
  - Current Score (numeric)
  - Average Score (system calculated)
- Register a new student profile
  - Student profiles are unique based on their First and Last name combination
  - First and Last name fields are therefore compulsory
  - Student Number needs to be derived from First and Last name combination
- Updating a student profile
  - The Update call should **only** allow the following fields to be updated
    - First Name
    - Last Name
    - Date of Birth
    - Cellphone Number
    - Email Address
  - Fields cannot be updated to an empty value if it already contains a value in the database.
- Capture a students' latest score
  - Student's score cannot be less than zero or more than 100
- Searching Student Profile
  - Users should be able to search by the following fields
    - Student Number
    - First Name
    - Last Name
    - Email Address
- Alerts (Success and Error) needs to be served by the Back-end.
- Apply alerts where needed.

## 4. Non-Functional Requirements

- The score progression/history should not be lost. The score history will be used in later versions. (Make sure it is stored in a usable way)
- The system must be robust and therefore requires validations.
- Users of the system will be using the UI as well as API integrations therefore these to systems need to act independently from each other.
- Documentation on the following is required:
  - Database setup and config
  - API setup and config
  - UI setup and config
  - API Integration documentation (including API calls, request and response examples and request validations)

## 5. Implementation

- The system must be distributed
- The system must have a persistent store for all the data
- The system must have unit tests
- SQL statements need to be used when accessing the database.

## 6. Bonus

- Functional programming principles applied throughout
- Use of Java Interfaces where possible
- Use of Immutability where possible
- Usage of Java 8 feature sets is encouraged

## 7. Additional Notes

- Use GitHub (or similar technology) – In order for us to pull the source code
- Use DockerHub (or similar technology) – In order for us to pull your docker images
- All services, where possible, needs to be dockerized.
- As far as possible compile all your questions in one email and send to us.

# 8. Appendix A - Wireframes

## 8.1 Search Student

Page 1

https://www.studentmanager.com/search

Student Manager

Search

Create

Quick Search

Search By \*

Search Criteria \*

Select

Search

Clear


Student No.	Full Name	Cell No.	Email	Current Score	Average Score	Action
E584598P	Ella Page	+7245678911	ella.page@gmail.com	87%	75%	<div>ADD SCORE</div> <div></div> <div></div>
C566854W	Chrissy Wolff	+7245672254	chrissyw@yahoo.com	66%	72%	<div>ADD SCORE</div> <div></div> <div></div>
V598755H	Valentina Hendrikx	+7245672356	valentina@outlook.com	99%	92%	<div>ADD SCORE</div> <div></div> <div></div>
L121245F	Lavinia Faure	+7245675684	lavinia242@gmail.com	27%	48%	<div>ADD SCORE</div> <div></div> <div></div>
C984576S	Cortney Sapienti	+7245677877	cspaggetti@gmail.com	74%	74%	<div>ADD SCORE</div> <div></div> <div></div>

<< Prev 1 2 3 4 5 6 7 8 9 10 Next >>

## 8.2 Create Student

Page 1

https://www.studentmanager.com/createstudent

 **Student Manager**

Search

Create

Create New Student

First Name \*

Last Name \*

Mobile Number \*

Country Code

Email Address \*

Date of Birth \*

Current Score

Save

Cancel