Registration Service

Group 6

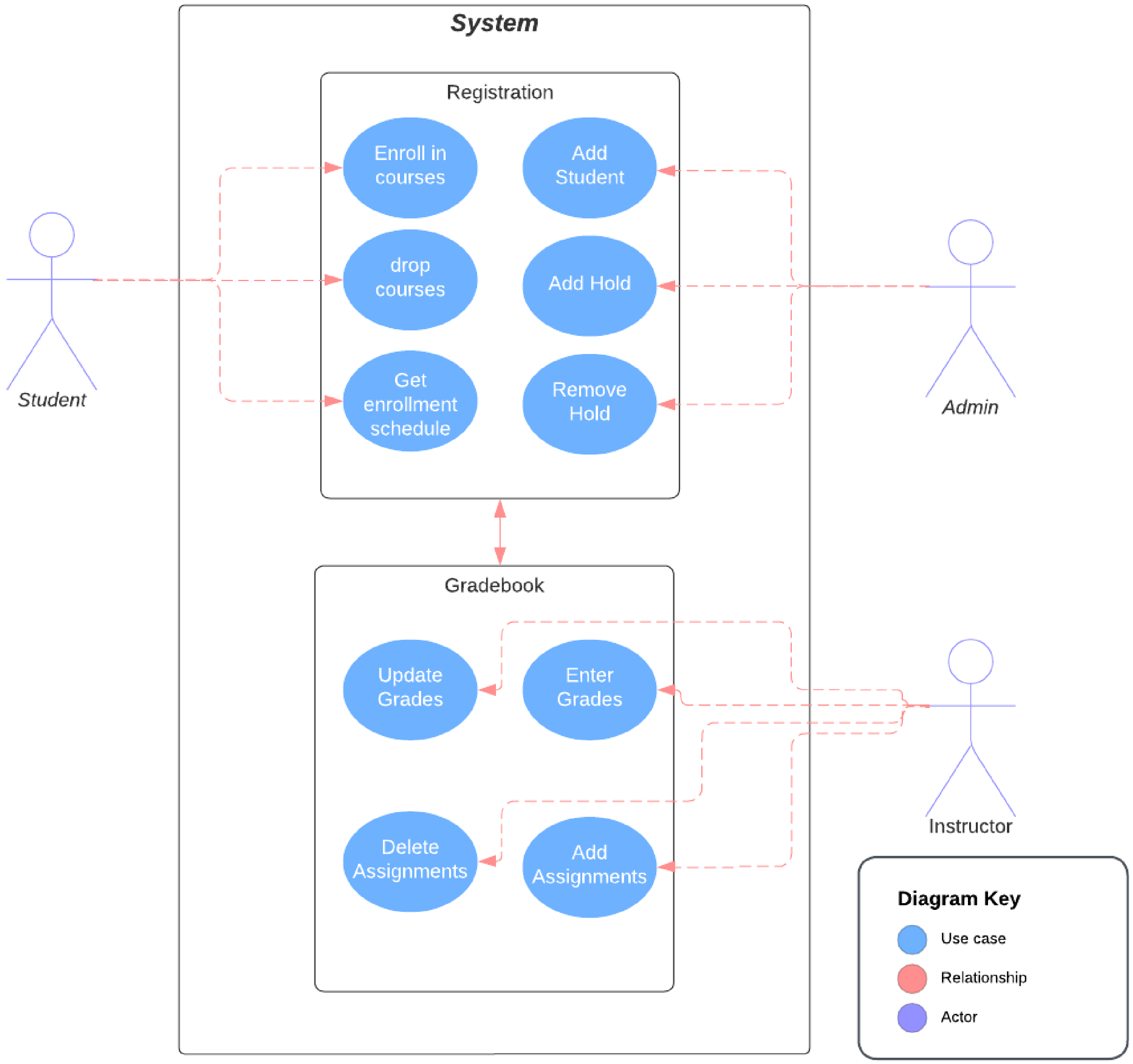
Yavik Kapadia & Saul Mendoza-Loera

Software Requirements Specification

Document

Version: 1 Date: 04/18/2023

1.1 System Environment



The learning management system has two services which have 3 actors and the services communicate with each other through a message queue. Students are able to add and drop courses, admins are able to add students as well as add or remove holds from student accounts, lastly Instructors are able to create, update,and remove assignments and grades. When students enroll in a course this enrollment information is passed to the gradebook service to update its enrollment table.

1.2 Functional Requirements

1.2.1 Administrator Use Case

Use Case Name	**Add a student**
Trigger	Admin presses Add student button
Precondition	Admin has to be signed in and on the index page
Basic Path	1. Admin clicks on **add student.** 2. Frontend presents admin with add student dialog. 3. Admin enters valid student email and name. 4. System checks where the student is in Database or not 5. If no student found, the system adds a new student to the database 6. Presents a success toast message to Admin
Alternative Paths	For step 3 if any of the information is missing an error toast will be presented
Postcondition	Student has been added and a success toast message is displayed
Exception Path	Admin may cancel adding new student
Other	

1.2.2 View schedule

Use Case Name	**Student Views Semester Schedule**
Trigger	Students select a radio button for a specific semester and presses **Get Schedule**
Precondition	Student has to be logged in and enrolled in courses for a semester
Basic Path	1. Students select the radio button for the semester they are trying to view the schedule for. 2. Then press get schedule 3. The system will get all the course the student is enrolled in for that particle

	semester and year.
Alternative Paths	None
Postcondition	Students are presented with this schedule.
Exception Path	Student may not click get schedule
Other	The schedule may be empty if the student is not enrolled in any courses

1.2.3 Enroll in a course

Use Case Name	**Student enrolls in a course**
Xref	1.2.2 view schedule
Trigger	Students select a radio button for a specific semester and presses Get Schedule
Precondition	Student has to be logged in and has no holds for registration
Basic Path	1. Student gets schedule for a specific semester they are cleared to enroll for 2. Student click on Add course button 3. Student enter valid course id 4. Student clicks Add button 5. On success student is presented with success toast message, dialog closes 6. Newly enrolled course is visible on schedule
Alternative Paths	At step 3, students may enter incorrect or invalid course id which will trigger an error toast message.
Postcondition	Students are presented with their schedule.
Exception Path	Students may not click get schedule. Students may select a semester they are not cleared to enroll in.
Other	

1.2.4 A student drops a course

Use Case Name	**Student drops a course**
Xref	1.2.2 view schedule, 1.2.3 Enroll in a course
Trigger	Student clicks **Drop** button next to a course
Precondition	Students have to be logged in and have courses enrolled in.
Basic Path	1. Students click the **view schedule** for a valid semester they can drop classes from. 2. Students then click the **drop** bottom next to a course. 3. Students are then prompted to click **cancel** or **ok** to confirm whether they want to drop course. 4. A success toast will be displayed upon success.
Alternative Paths	At step 3, student may click cancel to not drop the course.
Postcondition	Students are presented with their schedule without the course that was dropped.
Exception Path	Students may not click get schedule. Students may select a semester they are not cleared to drop courses from.
Other	Additionally if drop date has passed students may not be able to drop courses

1.3 Non Functional Requirements

1.3.1 Performance

Ideally the frontend and backend services should have multiple instances running across multiple servers and be behind a load-balance that can route requests to different instances depending on traffic. Considering that thousands of students will be accessing these services when class registration opens up thus our application needs to be able to handle thousands of requests at any given second.

1.3.2 Security

The registration backend should adhere to the following security requirements:

- Only authorized users should be able to access the API endpoints.
- Only authenticated admins are able to edit student information or add students.
- User authentication should be handled securely and should not be vulnerable to common attack vectors such as SQL injection or cross-site scripting.
- All communication between the client and server should be encrypted using HTTPS.
- The system should log all user actions for auditing and debugging purposes.
- The system should be regularly tested for vulnerabilities and weaknesses.

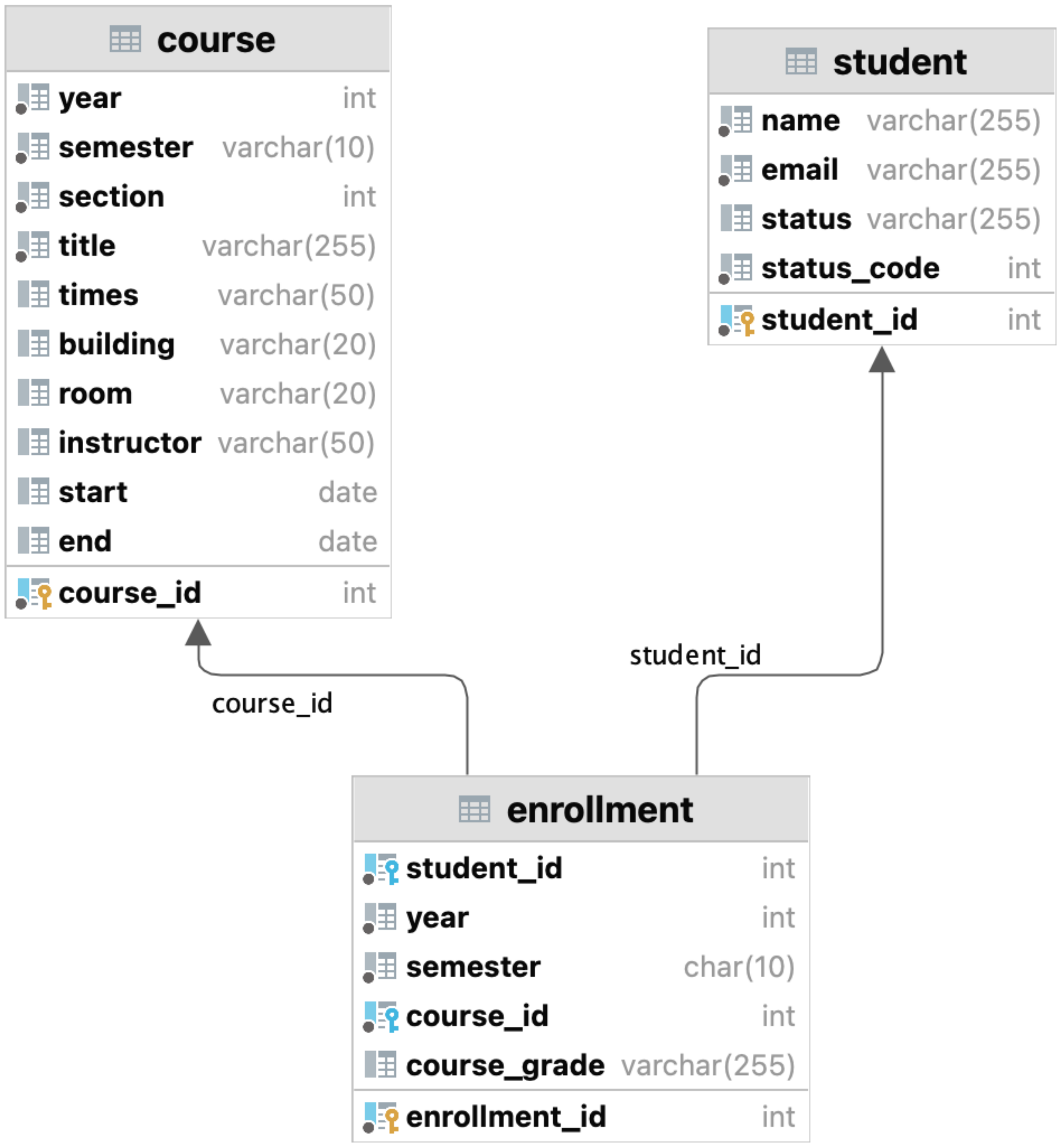
1.3.3 Browsers and Minimum requirements

The frontend of this service is designed using React which is designed and tested for recent mobile and desktop browsers, for touch and mouse and keyboard interactions.
The browsers with known support include:

- Google Chrome: version 80 and above
- Mozilla Firefox: version 72 and above
- Apple Safari: version 13 and above
- Microsoft Edge: version 80 and above

1.4 Logical database diagram and accompanying details on tables, keys, and attributes

1.4.1 Registration Database



1.4.2 Student Data Table

Student Data Table			
Data Item	**Data Type**	**Description**	**Comment**
student_id	int	Unique id for student	
name	varchar(255)	Name of student	
email	varchar(255)	Student email	
status	varchar(255)	Hold type, reason	
status_code	int	Integer code associated with hold type	

1.4.3 Course Data Table

Course Data Table			
Data Item	**Data Type**	**Description**	**Comment**
course_id	int	Unique id for course	
year	int	Year of semester	
semester	varchar(10)	Semester	Fall, winter, spring, summer
section	int	Section of course	Allows for multiple instances of a course
title	varchar(255)	Name of the course	
times	varchar(50)	Days and times course is being taught	
building	varchar(20)	building	
room	varchar(20)	Room in building	
instructor	varchar(50)	Instructor that is teaching a course	
start	date	Start date of course	
end	date	End date of course	

1.4.4 Enroll Data Table

Enrollment Data Table			
Data Item	**Data Type**	**Description**	**Comment**
student_id	int	Unique id for student	Foreign key from student table
year	int	Year of semester	
semester	varchar(10)	Semester	Fall, winter, spring, summer
course_id	int	Section of course	Foreign id from course table
course_grade	varchar(255)	Student grade	Grade reported from gradebook
enrollment_id	varchar(50)	Unique id for a specific students enrollment in course	