Software Design Specification

NUAPP

An application to ease access to Gate-pass, library, timetable and attendance services

CONTENTS

NUAPP

CONTENTS

Introduction
- Purpose of this document
- Scope of the development project
- Definitions, acronyms and abbreviations
- References
- Overview of the document

System Architecture Description
- Overview of modules / components
- Structure and relationships
  - Database Engine
  - Frameworks
  - Cloud Services
  - Server Application
  - Client Applications

Detailed Description Of Components
- Component template description
- Class Diagram:
- Description of components of Class Diagram:
  - 1. Student
    - 1.1 Attributes
    - 1.2. Operations
  - 2. Timetable
    - 2.1 Attributes
  - 3. Class
    - 3.1 Attributes
  - 4. Course
    - 4.1 Attributes
  - 5. Book
    - 5.1 Attributes
- Sequence Diagram

NOTE: In order to increase the understandability of the sequence diagram, it is broken into pieces according to functionality, but IN ORDER OF THEIR OCCURRENCE.
1. Login Function
2. Gatepass Functions
2.1 checkGatepassStatus( )
2.2 applyGatepass( )
3. Library Functions
3.1 getIssuedBookDetails( )
3.2 searchBook( )
4. Nucleus Functions
4.1 getTimetable( )
4.2 getAttendance( )
Reuse and relationships to other products
Design decisions and tradeoffs
Pseudo Code for components
Appendices (if any)

Introduction

Purpose of this document

This Software Design Specification (SDS) document will focus on specifying a high-level view of the architecture of our system, and on the interaction between the user and the system.
This document's purpose is to provide a high-level design framework around which to build NU APP . It also provides a list of requirements against which to test the final project and determine whether we were able to successfully implement the system according to design.

Scope of the development project

The purpose of the project is to integrate the three services that are Gate-pass, Library and the Nucleus of NIIT University in one place so that the user can access them more easily and conveniently. We are building a mobile application to help the students of NIIT University so that:

- The students should be able to view their current timetable with ease.
- Should be able to apply gate-pass through a mobile application itself.
- It must notify the student and concerned parent / guardian about the same.
- Should be able to view their issued books, to be notified about the reissue date.
- Student should be able to search for specific book availability in the library.

Definitions, acronyms and abbreviations

Words	Meaning / Definition
HTTP	Hyper Text Transfer Protocol
FTP	File Transfer Protocol
CRUD	Database basic operations: Create, Read, Update and Delete.
CLI	Short for Command Line Interface
Cordova	A platform which eases the process of hybyrid mobile application development by providing easy access to mobile APIs, and enhances the mobile development using HTML, CSS and JS.
KOHA database	The RDBMS used in our library systems.
HTML	Hyper Text Markup Language, used to write the web pages. Or in our case could be used to write App’s pages.
CSS	Cascaded Style Sheets, used to design or position the HTML elements. Very important to give style to our app.
JS	Java Script Used to add interactivity to web pages, used to reduce server load as well.
Firebase	Firebase is a google product which provides easy access and usage of APIs that help in authentication, notification and storage.

References

Not Applicable

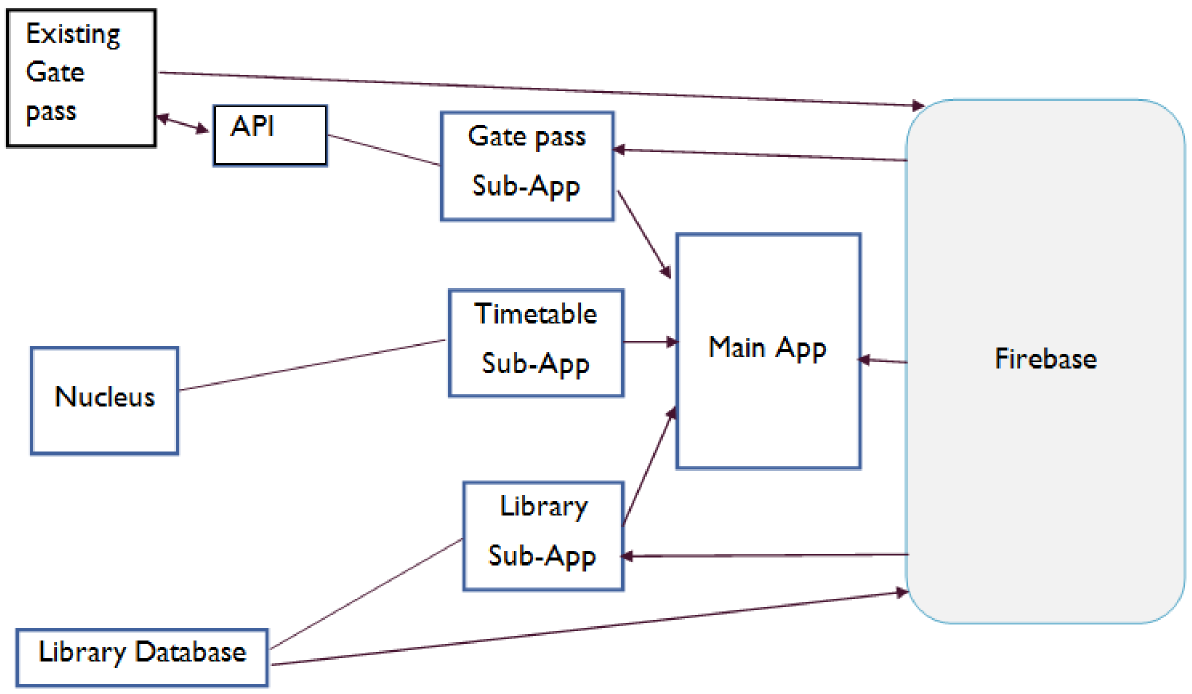
Overview of the document

System Architecture Description

Overview of modules / components

Our system is designed with the intention of making the application reliable,usable and robust. This application will fit into the daily lives of the recommended users with great ease.
This application in future may be updated as per the changes committed in it’s parent application.
There are five basic, logical components of the system: the Database Engine, Cloud service, the Server Application, and the Client Applications,Framework.

Structure and relationships



Database Engine

Hosts the backend database which is used for central data storage and retrieval.
The DB hosted in NIIT University server.

Frameworks

- Ionic helps web developers build great mobile apps and Progressive Web Apps in a way that felt just like building websites. That means Ionic focus on taking the standard HTML, CSS, and JavaScript you'd use to build a website, and help you turn it into mobile running application.

- Apps work across native and web environments, which helps to wield the true powers of the underlying native SDKs and device features.

- Angular reuse the code and abilities to build apps for any deployment target i.e, web, mobile web, native mobile and native desktop.

Cloud Services

- A **cloud service** is any resource that is provided over the Internet. The most common **cloud service** resources are Software as a **Service** (SaaS), Platform as a **Service** (PaaS) and Infrastructure as a **Service** (IaaS)
- FCM, can notify a client app that new email or other data is available to sync. FCM can send notification messages to drive user engagement and retention .For use cases such as instant messaging, a message can transfer a payload of up to 4KB to a client app.

Server Application

- Implemented in .NET
- Provides methods and procedures that can be invoked remotely by a client application via API calls.
    - Retrieve project data.
    - Update project data.
    - Generate reports.

Client Applications

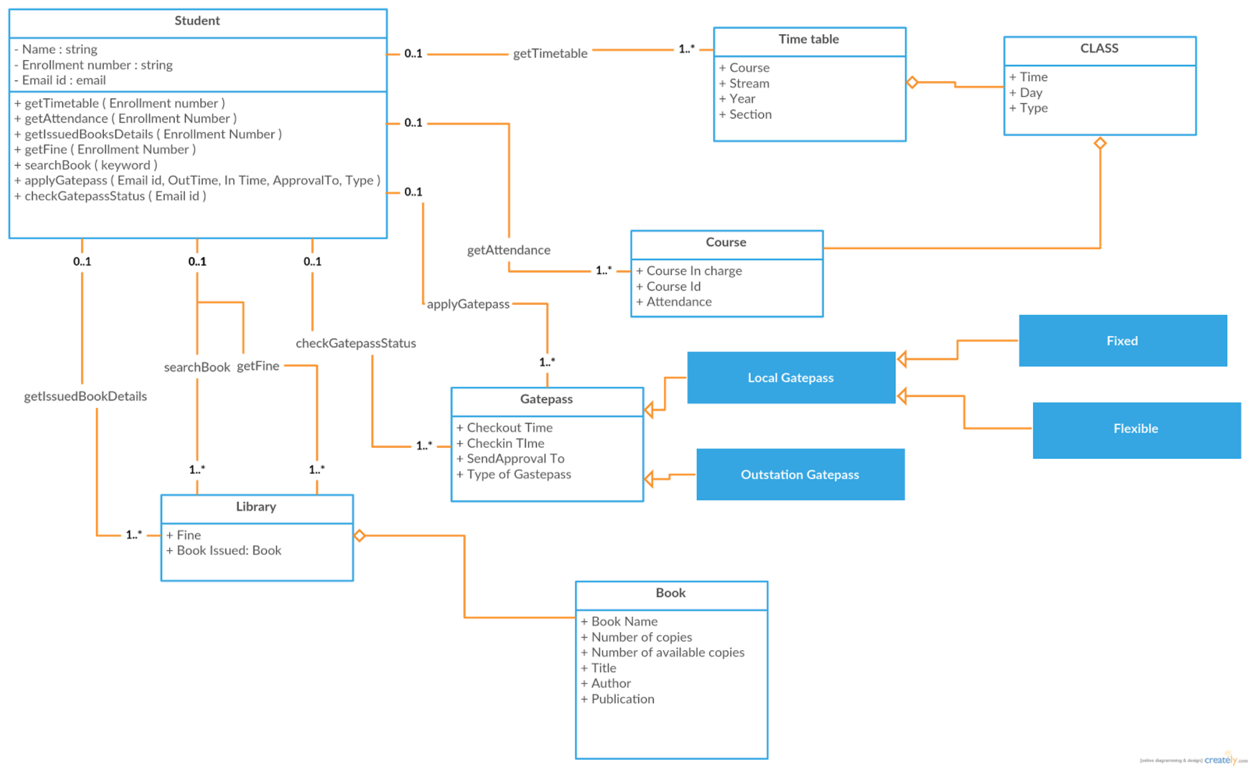
- Implemented in HTML5, SASS, JS, Anuglar-2, Ionic-2.
- Contains all presentation logic.
- Interacts exclusively with the user.
- Communicates with the server application through API's.

Detailed Description Of Components

Component template description

Following are the structures and relationships between various modules:

Class Diagram:



Description of components of Class Diagram:

1. 1. Student

1.1. 1.1 Attributes

Name	**Type**	**Description**
name	string	Name of the student
enrollment_no	string	Enrollment number of the student.
email_id	string	Registered email id of the student.

1.2. Operations

Name:	getTimetable( )
Input:	Student . enrollment_no
Output:	Returns a class object of type Timetable.
Description:	This function uses student's enrollment number to call the Nucleus' API to get the current timetable of the student, and return it by parsing it to a data object of Class type Timetable.

Name:	getAttendance( )
Input:	Student . enrollment_no
Output:	Adds an attribute attendance to the course class of each student.
Description:	This function uses student’s enrollment number to call the Nucleus’ API to get the current attendance of the student in each course, and then adds it to the course class for each subject.

Name:	getIssuedBookDetails( )
Input:	Student . enrollment_no
Output:	Returns a class object of type books.
Description:	This function uses student’s enrollment number to call the Library’s API to get the current books issued by the student, and return it by parsing it to a data object of Class type Books.

Name:	getFine( )
Input:	Student . enrollment_no
Output:	Returns a value which is the fine of the student.
Description:	This function uses student’s enrollment number to call the Library’s API to get the current fine of the student, and return it.

Name:	searchBook( )
Input:	Student . keyword

Output:	Returns a collection of objects of type Book.
Description:	This function uses keyword provided by the student to call the Library's API to get the books list, and return it by parsing it to a data object of Class type Book.

Name:	applyGatepass( )
Input:	Student . enrollment_no, Gatepass class
Output:	Returns a class object of type Gatepass.
Description:	This function uses student's enrollment number to call the Gatepass' API to apply a gatepass, and return the applied gatepass by parsing it to a data object of Class type Gatepass.

Name:	checkGatepassStatus( )
Input:	Student . enrollment_no
Output:	Returns a class object of type Gatepass.
Description:	This function uses student's enrollment number to call the Gatepass' API to get the latest gatepass of the student, and return it by parsing it to a data object of Class type Gatepass.

2. Timetable

2.1 Attributes

Name	**Type**	**Description**
Course	string	The course name. E.g. B.Tech, M.Tech.
Stream	string	The stream. E.g. CSE, ECE.
Year	int	The year of course end. E.g. 2018, 2019.
Section	string	The section of the student. E.g. S4, S1.

3. Class

3.1 Attributes

Name	**Type**	**Description**
Time	string	Time of the class. E.g. 10:30-11:30
Day	string	Day of class, e.g Monday
Type	string	Type of class. Lecture or practical.

4. Course

4.1 Attributes

Name	**Type**	**Description**
Course Name	string	Name of Course
Course Incharge	string	Course Incharge of the course.
Attendance	string	Attendance of the student in the course.

5. Book

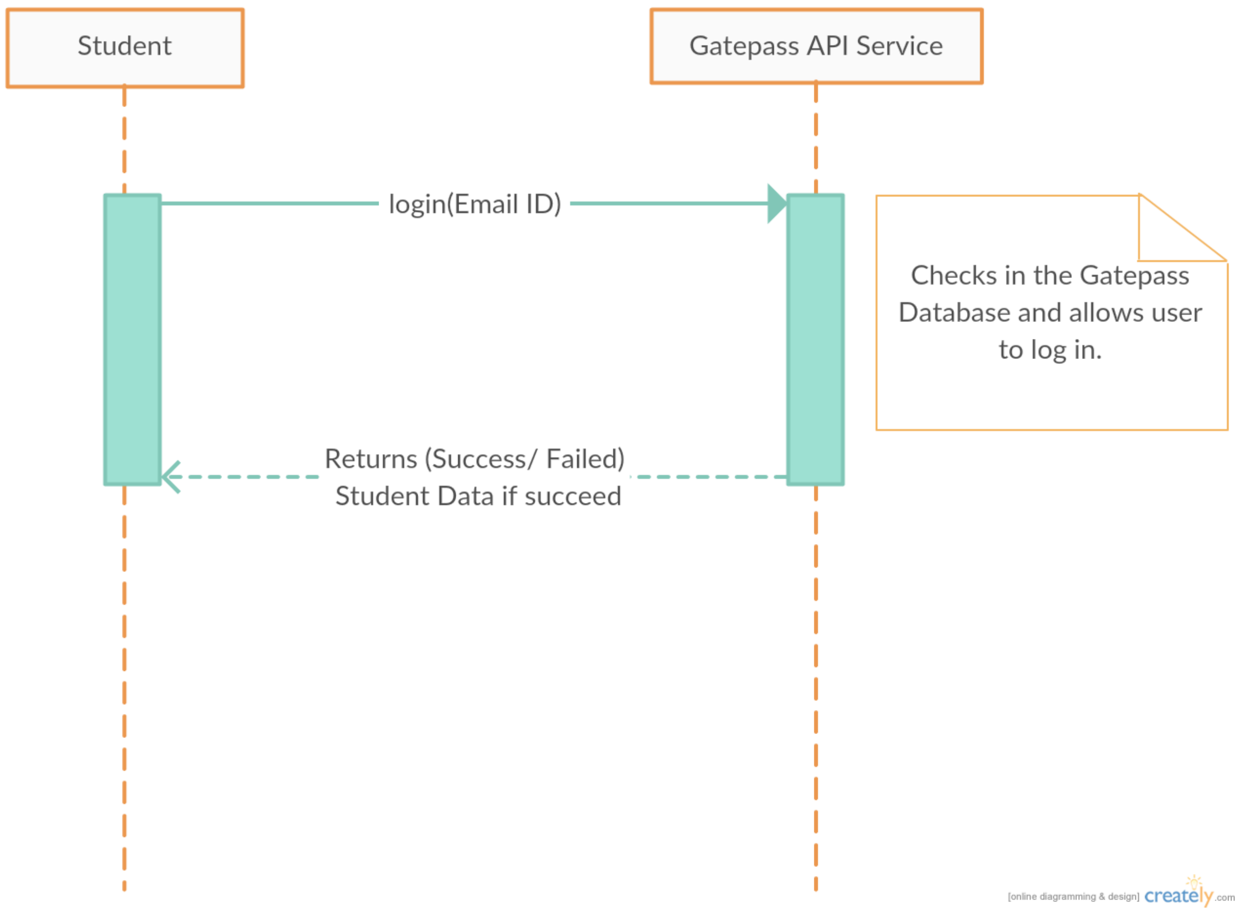
5.1 Attributes

Name	**Type**	**Description**
Title	String	Title of the book
Author	String	Author of the book
Publication	String	Publication of the book
Total_Copies	String	Total number of copies available.
Available_Copies	String	Number of copies available in the library.

Sequence Diagram

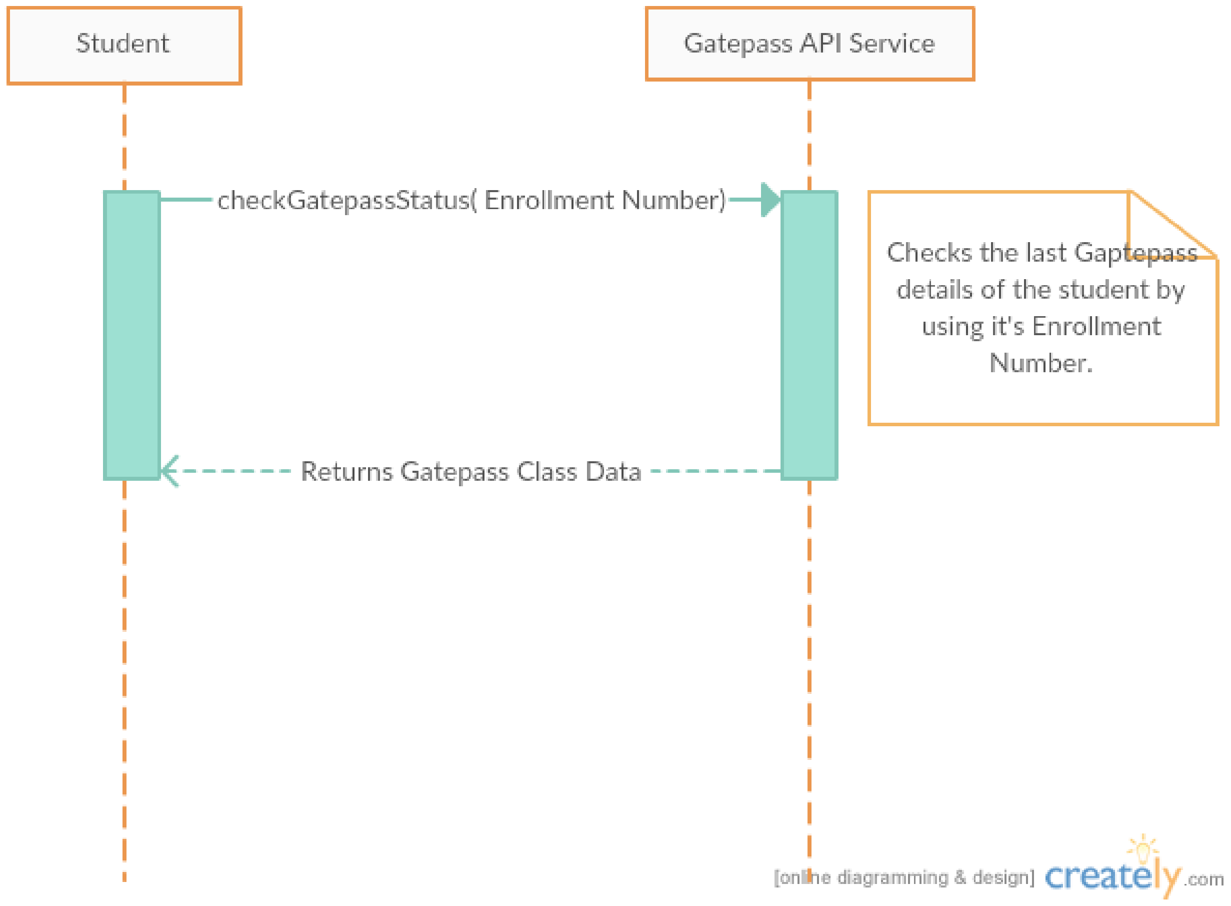
NOTE: In order to increase the understandability of the sequence diagram, it is broken into pieces according to functionality, but **IN ORDER OF THEIR OCCURRENCE**.

1. Login Function

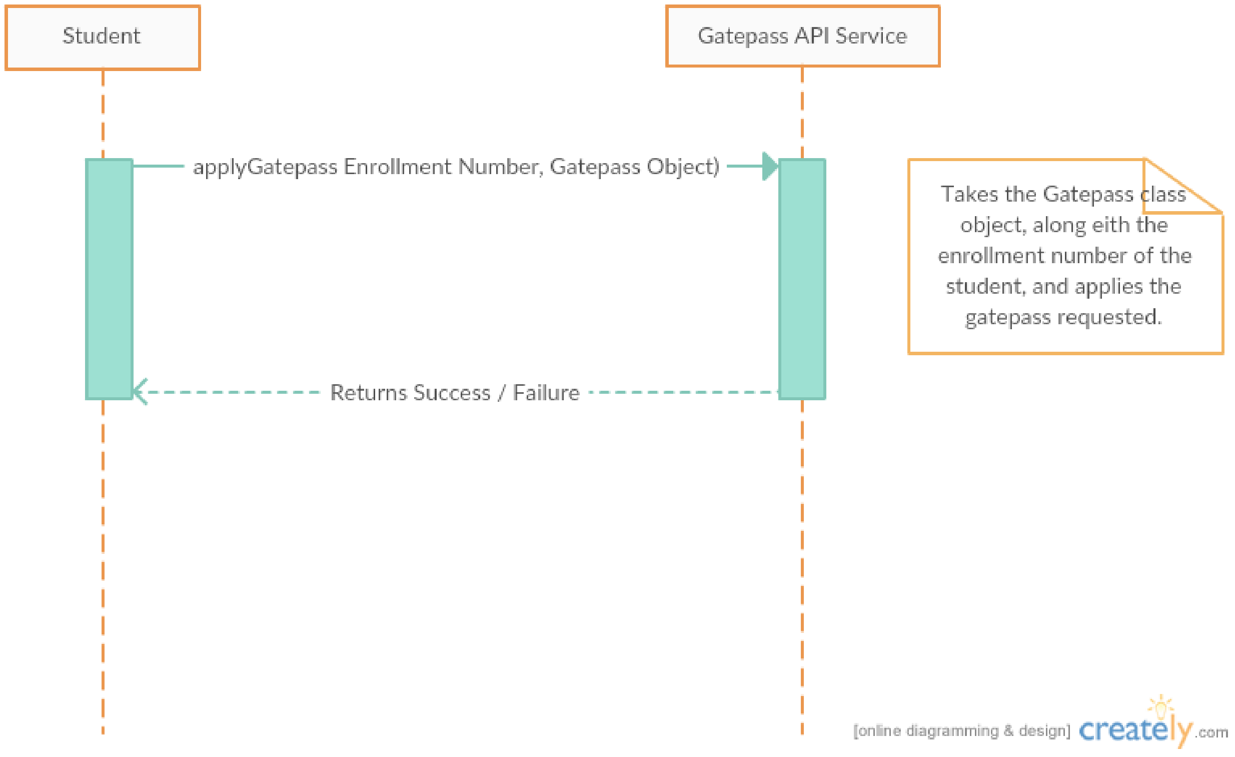

Student
Gatepass API Service
login(Email ID)
Checks in the Gatepass Database and allows user to log in.
Returns (Success/ Failed) Student Data if succeed
[online diagramming & design] creately.com

2. Gatepass Functions

2.1 checkGatepassStatus( )

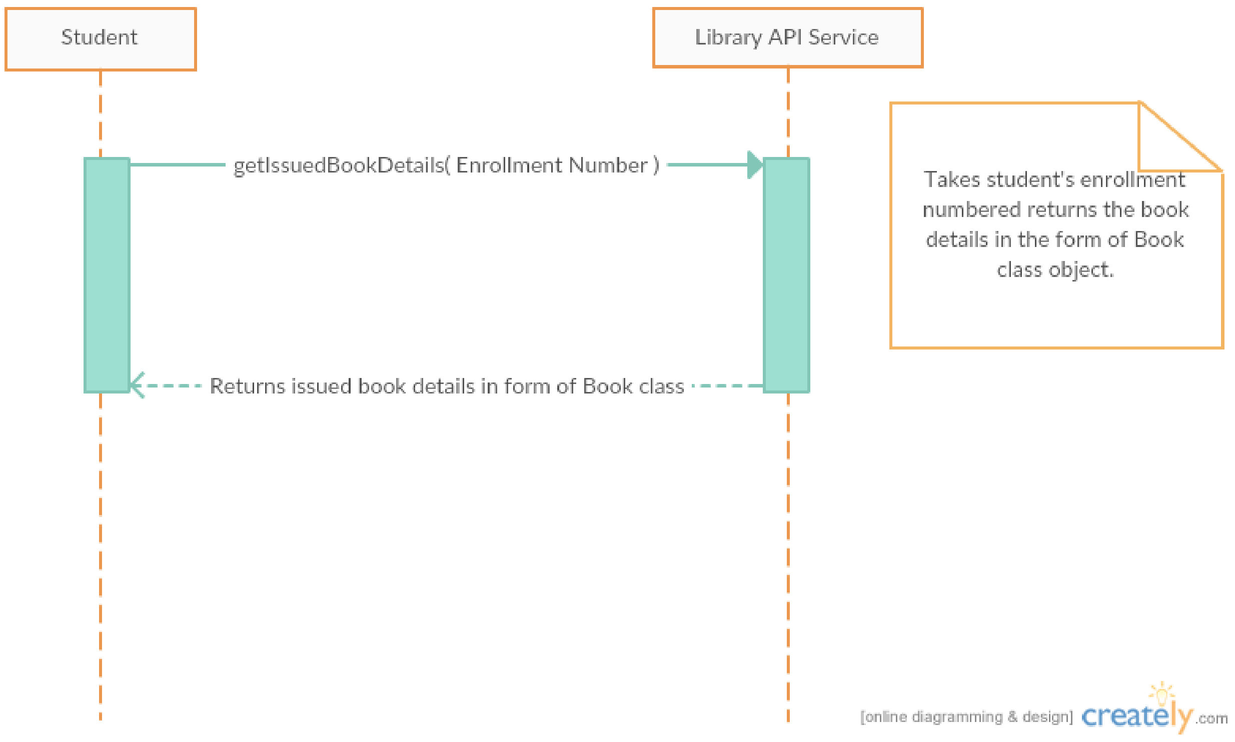

Student
Gatepass API Service
checkGatepassStatus(Enrollment Number)
Checks the last Gaptepass details of the student by using it's Enrollment Number.
Returns Gatepass Class Data
[online diagramming & design] creately.com

2.2 applyGatepass( )

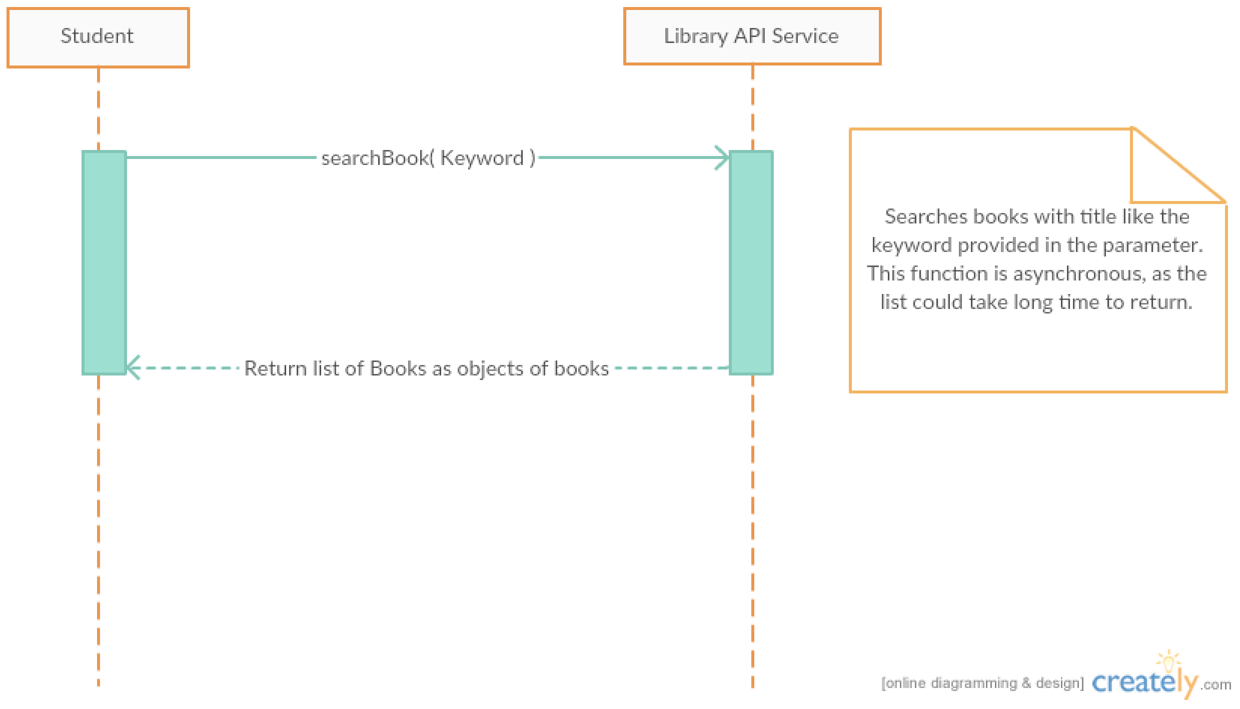

Student
Gatepass API Service
applyGatepass Enrollment Number, Gatepass Object)
Takes the Gatepass class object, along eith the enrollment number of the student, and applies the gatepass requested.
Returns Success / Failure
[online diagramming & design] creately.com

3. Library Functions

3.1 getIssuedBookDetails( )

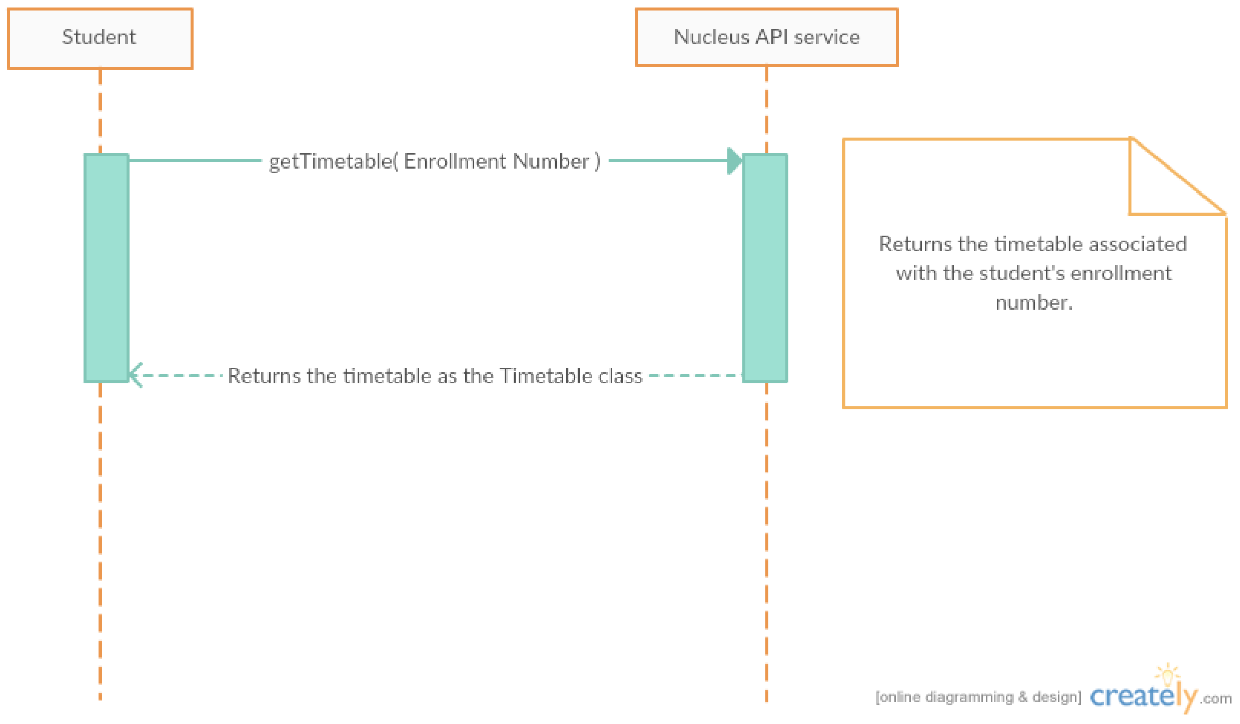

Student
Library API Service
getIssuedBookDetails(Enrollment Number)
Returns issued book details in form of Book class
Takes student's enrollment numbered returns the book details in the form of Book class object.
[online diagramming & design] creately.com

3.2 searchBook( )

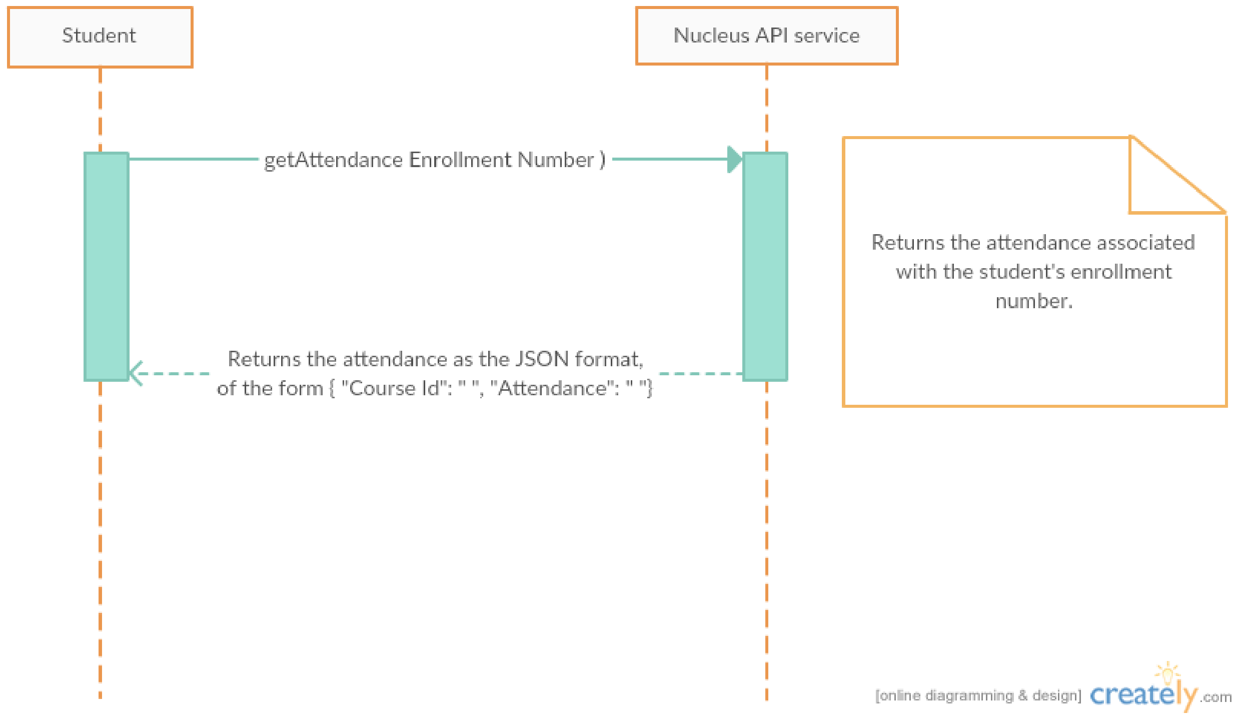

Student
Library API Service
searchBook(Keyword)
Return list of Books as objects of books
Searches books with title like the keyword provided in the parameter. This function is asynchronous, as the list could take long time to return.
[online diagramming & design] creately.com

4. Nucleus Functions

4.1 getTimetable( )


Student
Nucleus API service
getTimetable(Enrollment Number)
Returns the timetable as the Timetable class
Returns the timetable associated with the student's enrollment number.
[online diagramming & design] creately.com

4.2 getAttendance( )


Student
Nucleus API service
getAttendance Enrollment Number)
Returns the attendance as the JSON format, of the form { "Course Id": " ", "Attendance": " "}
Returns the attendance associated with the student's enrollment number.
[online diagramming & design] creately.com

Reuse and relationships to other products

N/A

Design decisions and tradeoffs

Design tradeoffs in the software are not only limited to the code and the application but also applicable to the device where the application resides.

Firstly, the methodology that we are using is Agile Scrum as this method of designing software is fast and also allows the developers to know where we are heading and it also enables to deliver the highest priority functionality first so to win the influence of the customer.

Secondly, the application will actually reside on the mobile phone of the users and will be using API's for further interaction, that will be installed on the NIIT University Servers.The application will only work if the client device is connected to NIIT University network.

Thirdly, the tradeoffs made during the development processes in software were made to increase performance and increase efficiency we are using ionic framework that provides the developers to use better components that are easy to implement and provides more power and independence as UI designers.Second great advantage of using Ionic 2 framework is that it's relatively easy to make an cross platform application in ionic 2 rather than using the traditional native way.
Ionic 2 framework using simple web technologies for creating a cross platform application.

Pseudo Code for components

```
Login(){
        If (user already exists) {
         // redirect him to the time-table page
        Goto timetable page
        }

        else {
        Input user email-id
        /* as a result the user will be added to the database and will directly be directed to
        the timetable page. */
        Remember user-email-id
        Goto timetable page
        }
}

get_Timetable(Student_enrollment_no) {

/* in this we pass the respective student's enrollment number and then the API return the
Class object that will be further parsed and will be provided to the end user. */

}



get_Attendance(Student_enrollement_no) {

/*whenever this function is called by the user it gives the attendance status of the
respective courses that the user is currently enrolled in.*/

}
```

```
get_IssuedBookDetail(Student_enrollement_no){

// assumptions whenever we issue a book our enrollment number is stored as the book-owner-id
       if(book-owner-id == enrollment number){
       Return book_name;
       }

       esle{
       Display "no book issued"
       }

}

get_Fine(Student_enrollment_no){

/* On calling this function the user's enrollment number is passed as an input to the Library
API which in-turn returns fine total amount of fine associated with it. */

}

SearchBook(char keyword){
// this function invokes the Library API to get the book list and return it by parsing it to a
data object of Class type Book.

Input the keyword
Search the keyword
Display the top 10 results.

}

ApplyGatepass(Student_enrollment_no, gate pass-type){

/*This function uses student's enrollment number to call the Gatepass' API to apply a
gatepass, and return the applied gatepass by parsing it to a data object of Class type
Gatepass*/

}


CheckGatepassStatus(Student_enrollment_no ){

/* Whenever this functionality is used by the user it calls the Gate-pass API and will return
the status of the recent activity by that user */

}
```

Appendices (if any)

N / A