# Software Design Document

# Digital Campus

| DATE | REVISION | REVISED BY |
|---|---|---|
| 8.08.2020 | Software Design Document Creation | Team-1 |
| 12.08.2020 | Software Design Document Revision 1 | Team-1 |

## 1. Introduction

### 1.1. Purpose

Digital Campus is a product designed for the ease of transition of higher education from an offline to an online platform. The product enables students to remotely access classes and labs conducted by their respective faculty through online mode. Setup of video/audio-based live theory classes supported with screen sharing, availability of recorded classes, and a virtual whiteboard for pointers. Attendance and schedule of classes are done through the system. Various labs can be carried online through animation, simulation, remote access to machines, or recorded videos of the demonstration of the experiment by the teacher.

### 1.2. Project Scope

Digital Campus has multiple tools for the administration of the institution to utilize in furthering communicating with their students – labs, automated attendance specifically designed for educational purposes. The proposed application takes care of both theories as well as lab classes and encourages comprehensive learning in all the fields.

### 1.3. References

Team 1, 2020. Product Requirement Document. Last modified : 28.07.2020

## 2. Design considerations

### 2.1. Assumptions

The user is aware of basic operations of a computer and web pages. The user also understands the standard terms used for operation.

## 2.2. Constraints

The required bandwidth and network should be met for efficient execution of the application. The users of the web application must be registered in the system to access its features.

## 2.3. System Environment

The web base online education system is designed to work on all operating systems with installed web browsers. It is accessible at all times.

## 2.4. Design Methodology

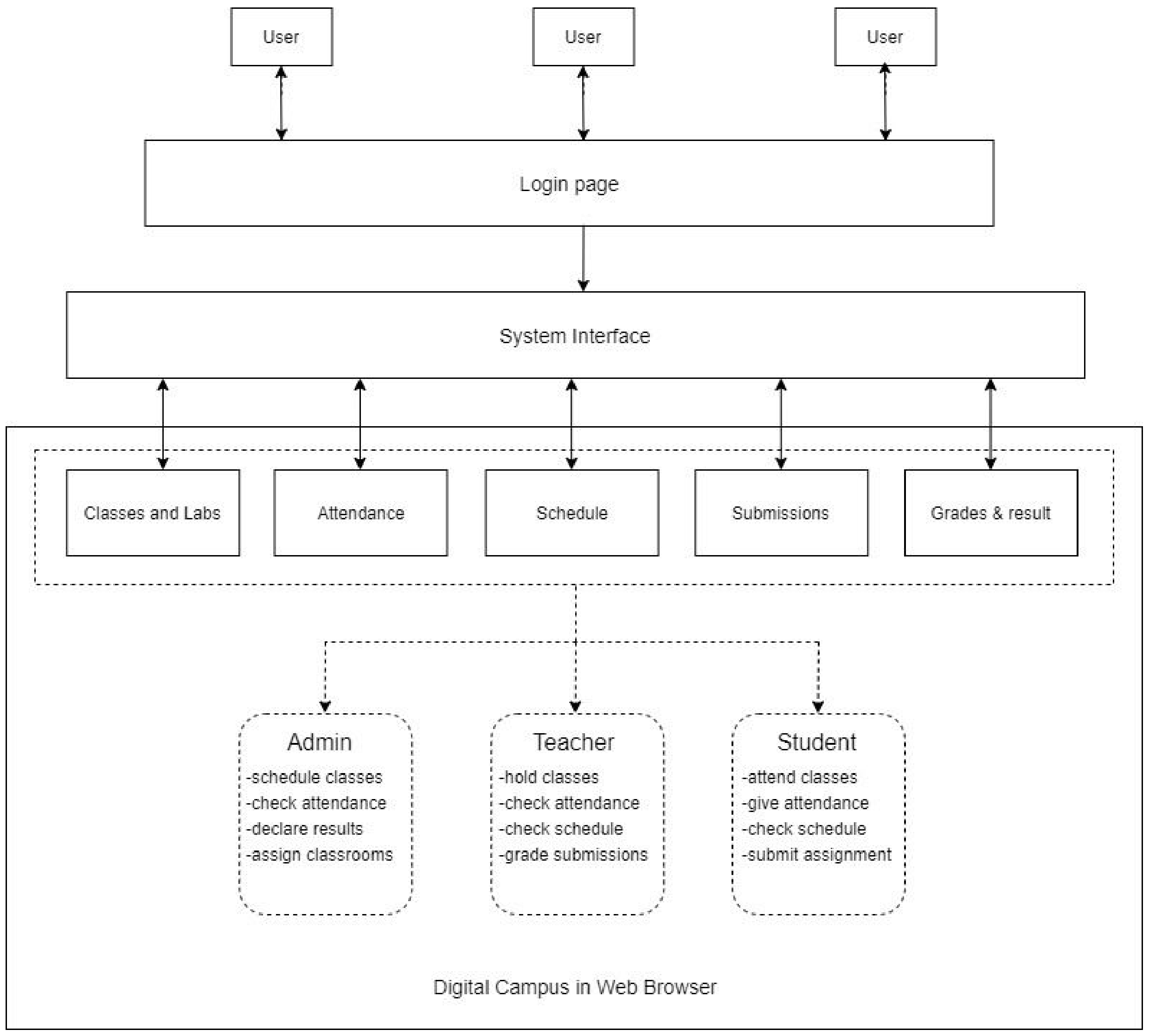
The system is designed with flexibility for further development and/or modification. Later versions of the software can be released with updates suggested by the user. The system is made adaptable to consider all types of users.

# 3. Architecture

## 3.1. System Design

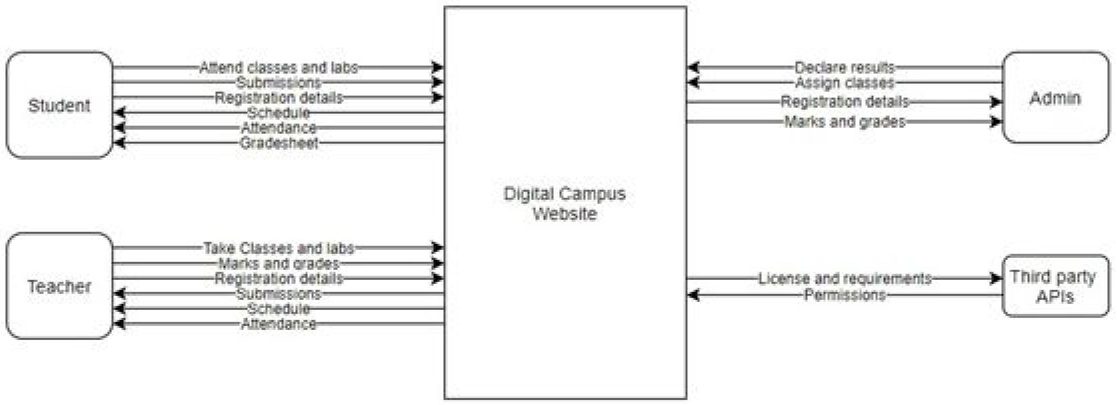
### 3.1.1. System block diagram

The block diagram below shows the principal parts of the system and their interactions.



### 3.1.2. System context diagram

The context diagram shows the main actors interacting with the system.

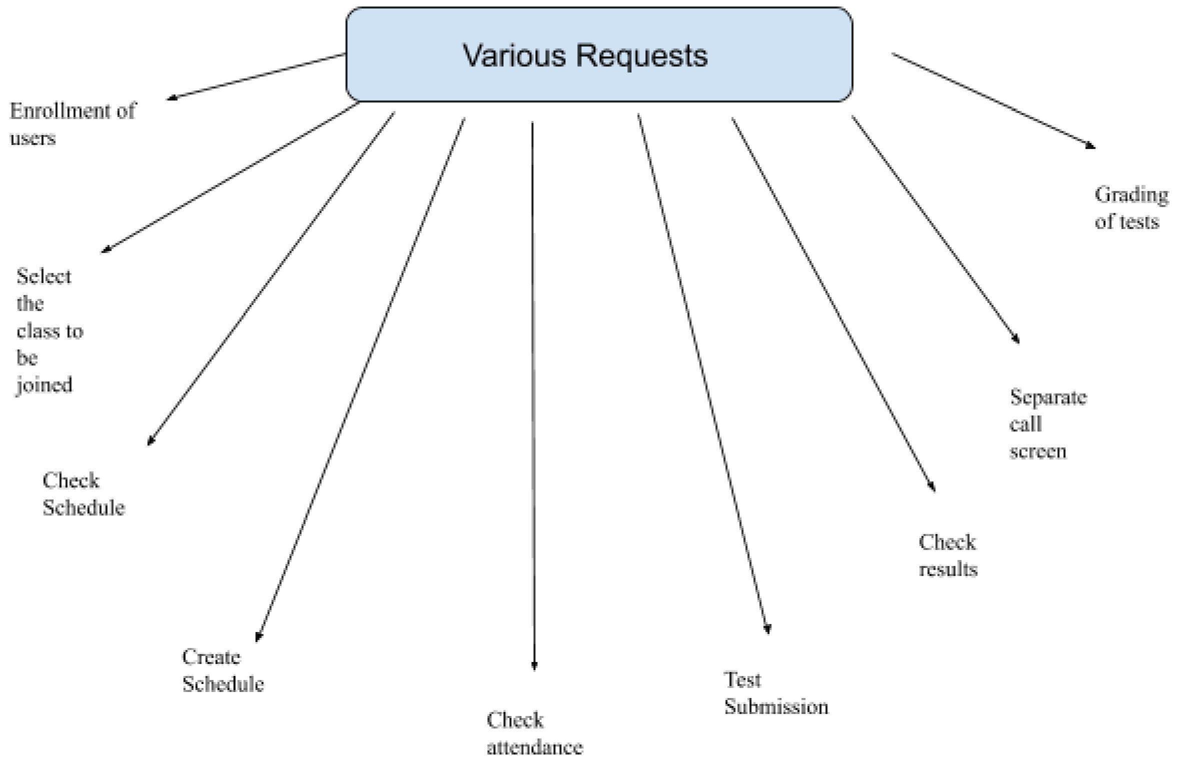


## 4. API Design

API design in order to document the way in which external consumers can understand how to work with code to get work done. RESTful web application program interface is preferred that runs across multiple platforms and improves scalability by simplifying the server components.

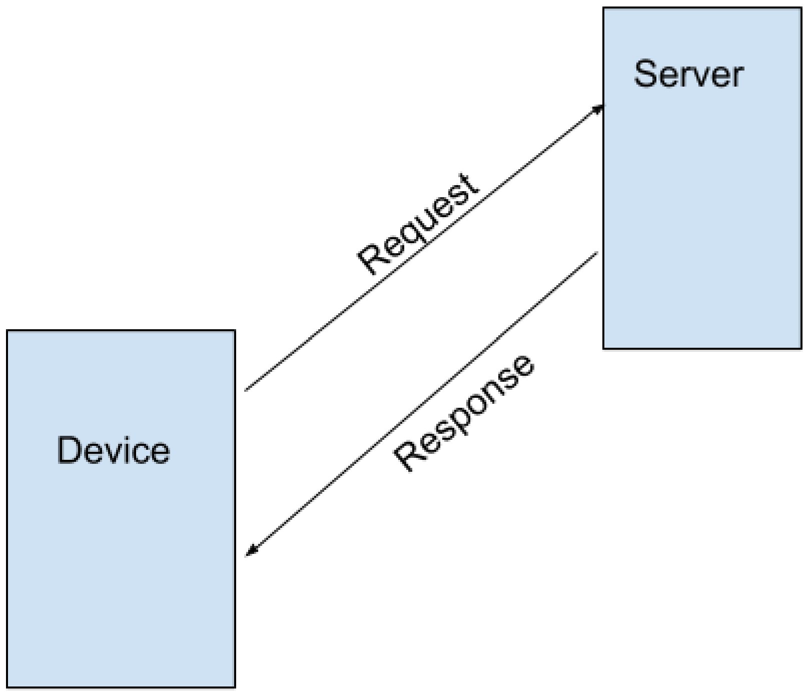
### 4.1. Allowed HTTP Requests

Get - Read data from a source
Post - Create a new resource, or perform an action
Patch - Update a resource with new values
Delete - Remove a resource
Put - Replace a resource with a new one



## 4.2. Resources

- Resources should be part of the resource model of the API that defines the data type and their behaviour.
- Resource models should interact with resources using ***methods***; for example, to send an email, use ***me/sendMail.***
- URL will include the resource you are interacting with in the request, such as ***me, user, group, drive and site.***
- Use JSON to model data as it is compatible with RESTful API and will provide the right balance between expressiveness and broad availability.
- Data associated with the resource is modeled as key:value pairs on the JSON object.
- In addition to application data, resources should also include other information that is specific to the RESTful API. Such information includes URLs and relationships.



Response is returned in json format.

## 4.3. User Server Responses

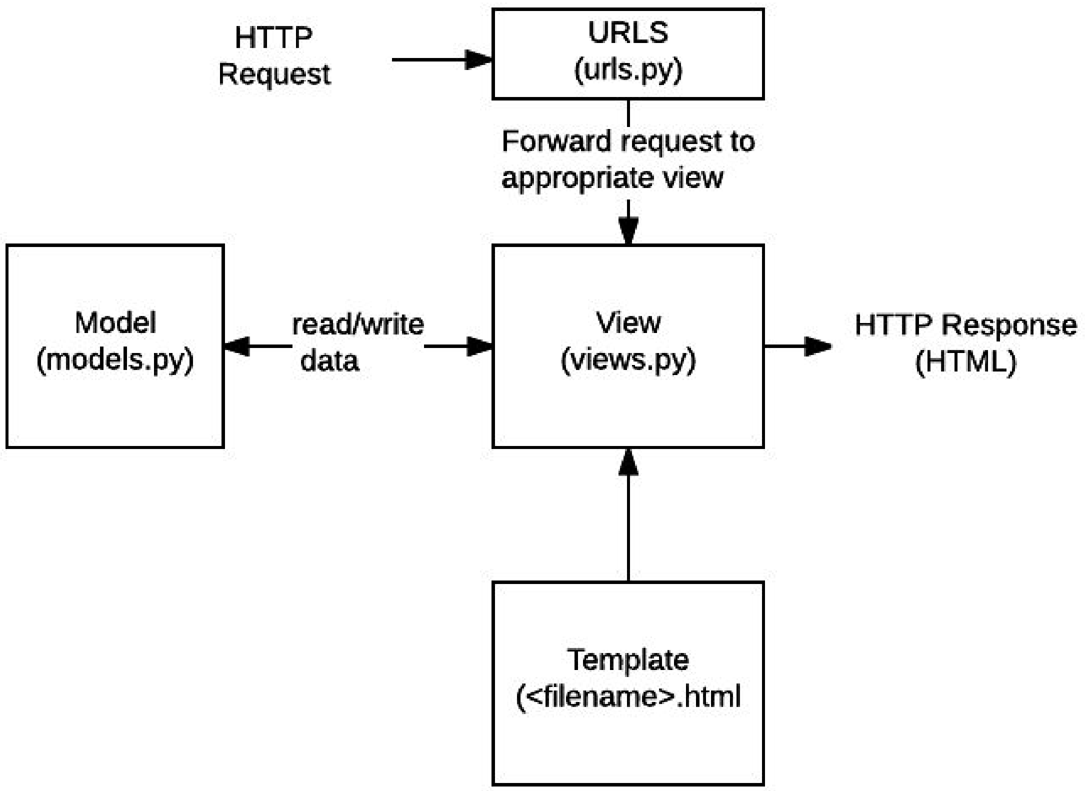
*OK* : The request was successful.
*Created* : The request was successful and a resource was created.
*No content* : The request was successful but there is no representation to return.
*Bad Request* : The request could not be understood or was missing required parameters.
*Unauthorised*: Authentication failed or user doesn't have permissions for requested operation.
*Forbidden* : Access denied.
*Not found* : Resource was not found
*Method not allowed* : requested method is not supported for the resource.

## 4.4. User Collection

- List all users - Students, faculty, administration.
- Create users - New users who opt in.
- Retrieve a user - Extract user information.
- Update a user - Scalable enough to bring in the update for all users.
- Remove a user - When a user opts out.

## 4.5. Framework(Django-Python)

Various functionalities that are necessary for optimising the application include:
Versatility, Security, Scalability, Maintainability and Portability. The application is implemented on DJANGO framework as it meets the requirements of the application and is also compatible for video applications.
The code format:



**Protocol used for video, audio and real time data** :- ***webRTC***
The APIs with this protocol include :

- ***getUserMedia API*** - This will provide access to multimedia streams from local devices. It can be used to record lessons for online class. Compatible browsers are chrome, firefox and opera.
- ***RTCPeerConnection API*** - This will allow an application to establish connection between the users.
- ***RTCDataChannel API*** - This enables peer to peer exchange of arbitrary data, with low latency and high throughput. It can be used to transfer files and for real time chat.

## 5. Performance Requirements

- Response time: Ideally should be 0.1 seconds (instant response, no interruption). Maximum limit is 1 second response time. If more than that, then there should be a dialogue box to inform the users of how much expected time it'll take for the site to load.
- Assumptions:

  1. The campus size is roughly 30,000, with ≈ 23,000 students and ≈ faculty members.

  2. Required browsers are available.

  3. Required version of the operating system is available.

  4. Any supporting utilities required to run the software are available.

- Workload:

| **Scenario** | **Daily total** | **Pages** |
|---|---|---|
| Attend classes | 30,000 | Login, Home, {Subject Name} |
| Attend labs | 20,000 | Login, Home, {Subject Name} |
| Submit tests | 23,000 | Login, Home, {Subject Name}, Assignment |
| View attendance | 30,000 | Login, Home, {Subject Name}, Attendance |

| | | |
|---|---|---|
| View schedule | 30,000 | Login, Home, {Subject Name}, Calendar |
| View results | 30,000 | Login, Home, {Subject Name}, Choose test |
| Grading tests | 700 | Login, Home, Student Details, Choose test |
| Create schedule | 200 | Login, Home, Schedule class |

- Scalability: the software has to be extendable to 5 other campuses; should be economically viable, response time requirements should still be met.
- Platform:

    1. Supported browsers:

        a. Windows: Chrome 30+, Edge 12+, Firefox 27+

        b. Mac: Safari 7+, Firefox 27+, Chrome 30+

        c. Linux: Firefox 27+, Chrome 30+

    2. See if any 3rd party software utilities needed

    3. Hardware requirements on Windows:

        a. Computer and processor: 32-bit or 64-bit; minimum 1.6 GHz (or higher), and a dual-core processor

        b. Memory: 2 GB RAM

        c. Hard disk: 3.0 GB of available disk space

        d. Display: 1024 x 768 screen resolution

        e. Graphics hardware: Minimum of 128 MB graphics memory

        f. Operating system: Windows 10, Windows 8 or 8.1, Windows 7, Windows Vista with SP1 or later, Windows XP with SP3 or later

g. Video: USB 2.0 video camera

h. Devices: Standard laptop camera, microphone, and speakers

4. Hardware requirements on Linux:

   a. Computer and processor: 32-bit or 64-bit; minimum 1.6 GHz (or higher), and a dual-core processor

   b. Memory: 2 GB RAM

   c. Hard disk: 3.0 GB of available disk space

   d. Display: 1024 x 768 screen resolution

   e. Graphics hardware: Minimum of 128 MB graphics memory

   f. Operating system: Ubuntu 12.04 or higher, Mint 17.1 or higher, Red Hat Enterprise Linux 6.4 or higher, Oracle Linux 6.4 or higher, CentOS 6.4 or higher, Fedora 21 or higher, OpenSUSE 13.2 or higher, ArchLinux (64-bit only)

   g. Video: USB 2.0 video camera

   h. Devices: Standard laptop camera, microphone, and speakers

5. Hardware requirements on MAC OS:

   a. Processor: minimum Intel processor, Core 2 duo (or higher)

   b. Memory: 2 GB RAM

   c. Hard disk: 1.5 GB of available disk space

   d. Display: 1280 x 800 screen resolution (or higher)

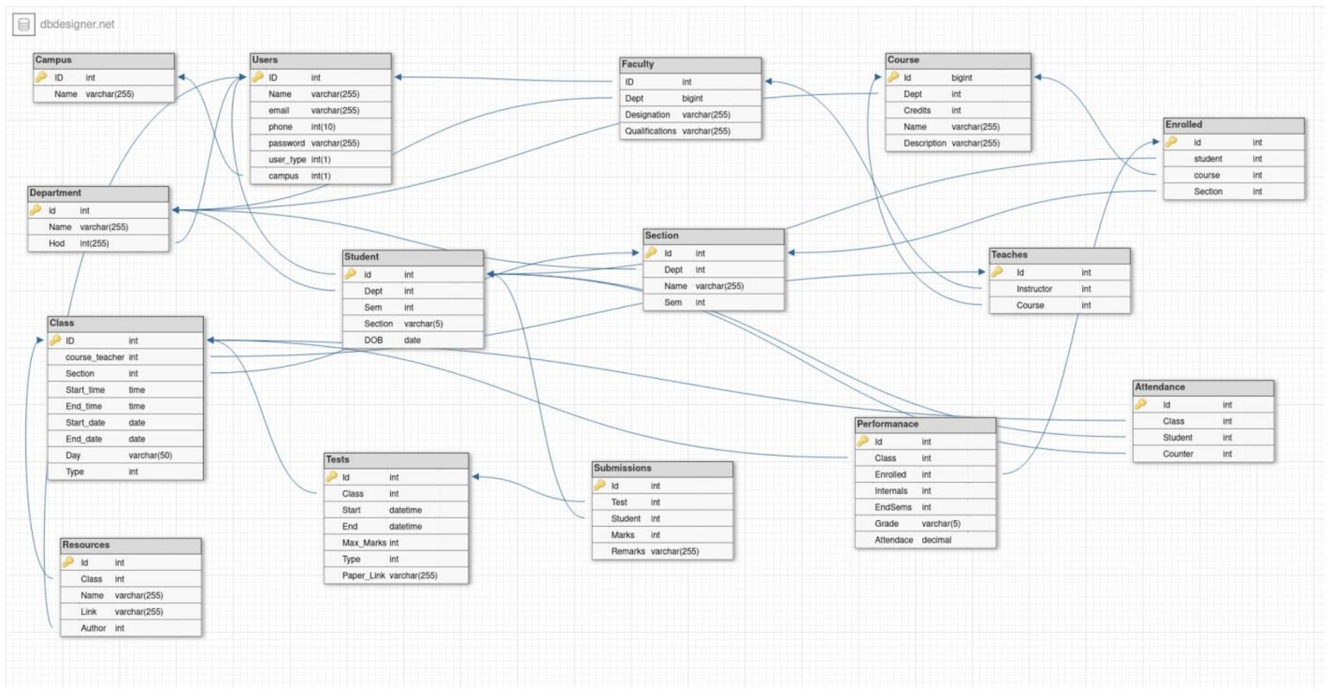
   e. Graphics hardware: Minimum of 128 MB graphics memory

   f. Operating system: Mac OS X with MacOS 10.6.8 /(Snow Leopard) or later

   g. Video: compatible webcam

h. Voice: Compatible microphone and speakers, headset with microphone, or equivalent device

## 6. Database Schema(For MySQL Database)



## 7. Software Interface Design

The designs of the UI can be found at- UI_Designs

## 8. Value Added Services

1. Library: Users of the university/colleges will be given access to their respective digital libraries and databases.
2. Customised user themes(for  classes and lobby) and sounds.
3. Frequent caller stats.
4. Weekly/monthly/daily usage stats.
5. Leaderboards(based on submissions/marks etc).
6. Early bird kickbacks(earliest submissions on average gives additional perks/unlocks new themes).
7. Student performance stats, trends.
8. Stats on teachers based on student performance.
9. Average time taken for submitting assignments/tests.
10. Average attention span of a student (based on the time taken to click the attendance pop up).