Section 1 - Project Description

1.1 Project

Virtual Authoring Tool

1.2 Description

2020 was a challenging year. The normal classroom experience was converted into a virtual environment. Certain classes that included labs became difficult to complete due to lack of hands-on activities. Students were not able to comprehend course material which caused an educational roadblock to passing a class. Virtual Lab is a cross-platform application that will provide an opportunity to allow students as well as instructors to gain back the experience of being in a real classroom.

1.3 Revision History

Date	Comment	Author

Contents

Section 2 - Overview

2.1 Purpose

Brief description of the focus of this module of the overall project and its intended audience.

- The purpose of this authoring tool is to allow all types of professors to create highly customizable virtual lab environments for their students to interact with. Students would be able to put their theoretical knowledge into practice with the created environments. The authoring tool contains various types of 3D objects that can be placed to fit the needs of the specific lab/environment.

2.2 Scope

Describe the scope of the module to be produced

- Professors should be able to upload 3D object files for the lab environment
- Students will be able to experience the 3D environment
- Students will also be able to interact with the tools and apply course concepts

2.3 Requirements

Your mileage may vary -- we typically break down the requirements to provide a ballpark estimate.

2.3.1 Estimates

#	Description	Hrs. Est.
1	Setting up A-frame environment & AWS services	# est
2	Upload 360 images, 2D video (lab instructions) into the environment	30
3	Object alignment & editing	15
4	Use pre-designed environment templates	15
5	User sign up and sign in	10
6	Create/Spawn 3D Animate objects	30
7	Upload audio for narration	10
8	Save environment templates	15
	TOTAL:	125

2.3.2 Traceability Matrix

Cross reference this document with your requirements document and link where you satisfy each requirement

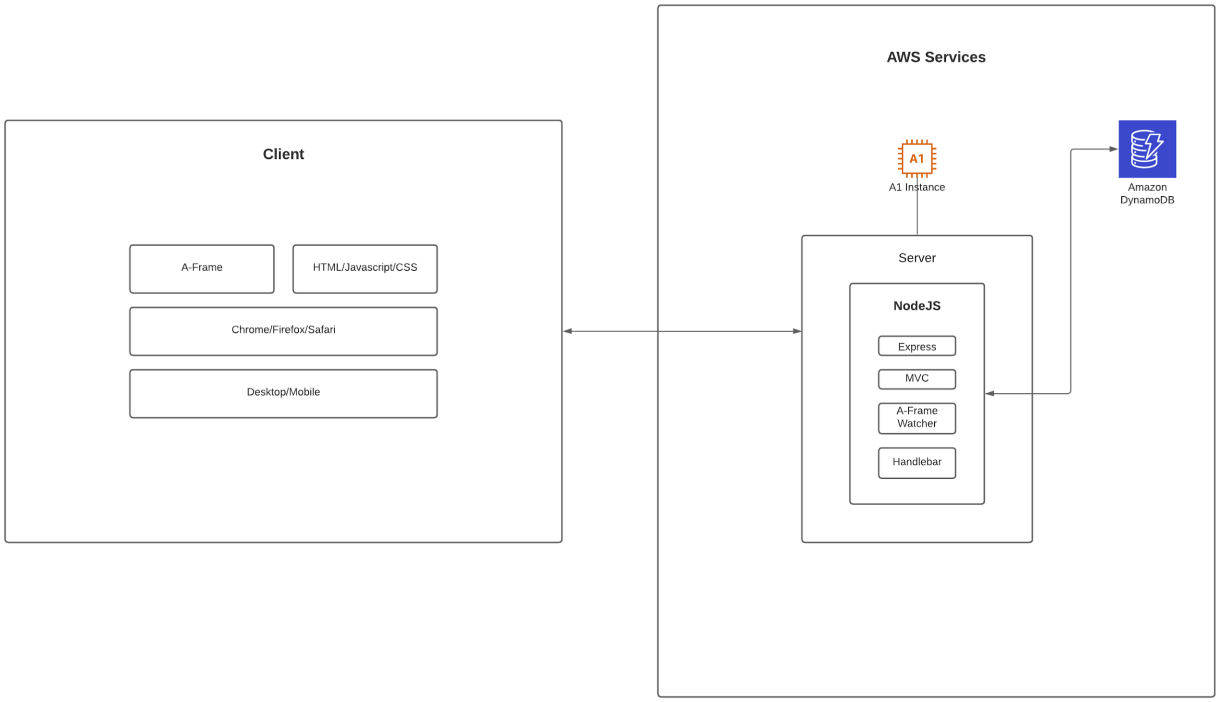
SRS Requirement	SDD Module
FR1: User registration • A user will register as either a student or professor • Each user will register with an email and password • Passwords are hashed before stored	5.2.1 User Registration

FR2: User login ● Professor ○ Will be able to set up virtual lab environments ○ Will be able to upload images, videos, and audio learning tools ○ Professors will have a dashboard ● Student ○ Will have a dashboard ○ Will be able to see the virtual lab environment set up by the professors ○ Will be able to interact with environment objects	**5.2.2 User Domain of Login**
FR3: Dashboard ● Professor ○ Display class info ■ Course name ■ Building name & room number ■ Student count ■ Publish/Update/Delete button ○ Create new virtual environment button ● Student ○ Display class info ■ Course name ■ Grade ■ Assignments ■ Discussions ■ Labs	**5.2.2.2 Professor Dashboard of User Domain** ***5.2.2.1 Student Dashboard of User Domain***
FR4: Image upload ● Images can be 2D or 3D ● The images will be pictures of any lab tool that will be used in the virtual lab environment	*5.2.2.2.4.1.1 Upload lab environment image*
FR5: Add/Edit object or description ● Professors will have the option to add a new object via image upload ○ During image upload the professor will be able to add a	5.2.2.2.4 Create and Edit Staff Dashboard of User Domain

description explaining the purpose of the object • The professor will have the option to edit any existing objects ○ The professor can change the object's description ○ The professor can add or remove an object from an environment	5.2.2.2.2 Upload the Lab Equipments of Professor Dashboard
FR6: Video/Audio upload • Professor will be able to upload video or audio recordings and insert it into their classroom environment • Video & audio is used for aid or informative guides or instructions • Audio can play when a student clicks on a pin mark for a certain object in the environment to help identify and elaborate its purpose	5.2.2.2.3 Upload Video and Audio of Professor Dashboard
FR7: Initiate Class Environment Template • Professors will have the option to create a new virtual class environment ○ Will be able to arrange the objects of the environment ○ Will be able to authorize which tools can be interacted by the students	*5.2.2.2.4.1 Environment of Create and Edit of Professor Dashboard*
FR8: Object/Environment Interaction • Pin mark clicked on ○ Written message ○ Additional information ○ Audio message • 3D render objects can be viewed ○ When clicked object will rotate	***5.2.2.3.2 Interaction of Attend Live Lab of User Domain***
FR9: Save Class Environment Template • After a professor is finished arranging and setting up the virtual environment, they have the option to save the setup ○ The virtual environment setup can be saved as a template ○ A saved environment template will be able to be reused	*5.2.2.2.4.1.4 Save Environment as a Template*

Section 3 - System Architecture

Describe/include a figure of the overall system architecture (and where this module fits in)



Section 4 - Data Dictionary

Brief description of each element in this module or a link to an actual data dictionary

Users		
Field	Notes	Type
id	Auto generated by database	DECIMAL
first_name	User's first name	STRING
last_name	User's last name	STRING
email	User's email	STRING
courses	List of user's courses	ARRAY
user_type	User's type either student or professor	STRING

updated	Timestamp of update done to user information	TIMESTAMP

Courses		
Field	Notes	Type
id	Auto generated by database	DECIMAL
title	Course's title	STRING
professor	Professor's full name	STRING
email	Professor's email	STRING
description	Brief description of the course	STRING
labs	List of labs available for the course	ARRAY
updated	Timestamp of update done to course information	TIMESTAMP

Labs		
Field	Notes	Type
id	Auto generated by database	DECIMAL
title	The title of the lab	STRING
grade_percentage	The grade percentage earned for the lab	FLOAT
description	Brief description of lab	STRING
environment_url	Path	STRING
completed	Set boolean value if lab was completed or not	BOOLEAN
updated	Timestamp of the update done to the lab information	TIMESTAMP

Object Tools		
Field	Notes	Type
id	Auto generated by database	DECIMAL
name	The name of the tool	STRING
url	The image path	STRING
description	Brief description of tool	STRING

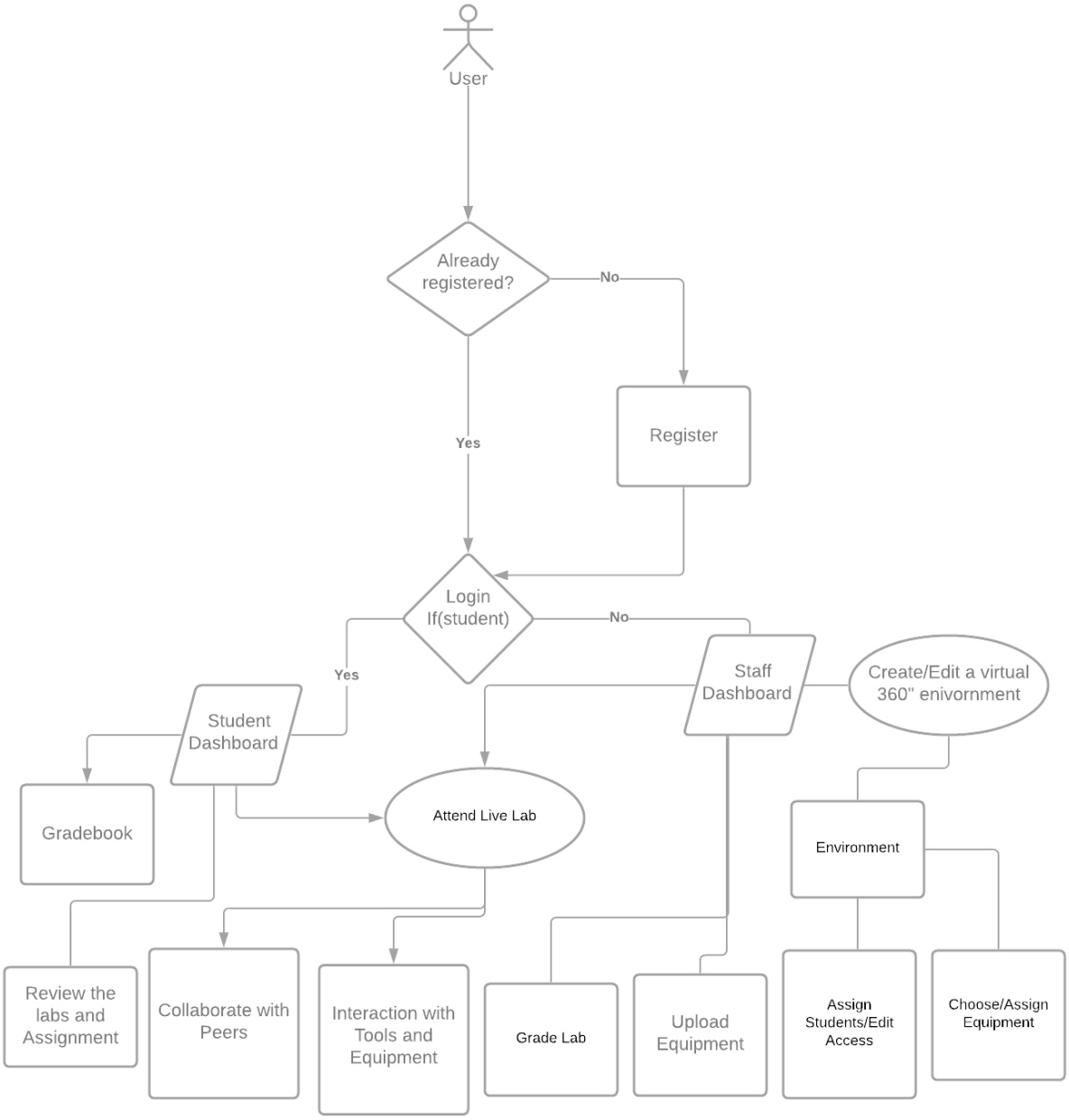
Videos		
Field	Notes	Type
id	Auto generated by database	DECIMAL
name	The name of the video	STRING
url	The video path	STRING
description	Brief description of tool	STRING

Audio		
Field	Notes	Type
id	Auto generated by database	DECIMAL
name	The name of the audio	STRING
url	The audio path	STRING
description	Brief description of tool	STRING

Section 5 - Software Domain Design

5.1 Software Application Domain Chart

Describe / chart each major software application domain and the relationships between objects (UML, etc)



[Link to LucidChart]

5.2 Software Application Domain

A Comprehensive high level description of each domain (package/object wherever it is better to start) within the scope of this module (or within the greater scope of the project if applicable)

5.2.1 User Registration

For users to be able to register.

5.2.2 User Domain of Login

Logs in the user and checks if it's a student with a bool function.

5.2.2.1 Student Dashboard of User Domain

If login (if Student) is true, it takes the user to a student dashboard.

5.2.2.1.1 Grade Book of Student Dashboard of User Domain

Students are able to access the gradebook to view graded assignments.

5.2.2.1.2 Lab Review of Student Dashboard of User Domain

Students are able to review labs and assignments.

5.2.2.2 Professor Dashboard of User Domain

If login (if Student) is false, it takes the user to a staff dashboard.

5.2.2.2.1 Lab Grading of Professor Dashboard

Based on the students lab submission, the professor will be grading the lab.

5.2.2.2.2 Upload the Lab Equipments of Professor Dashboard

Professors will be uploading the lab equipment on their dashboard to assign when the lab is posted.

5.2.2.2.3 Upload Video and Audio of Professor Dashboard

Professors are able to upload videos and audios that would be used in a lab environment.

5.2.2.2.4 Create and Edit Staff Dashboard of User Domain

Professor can create and edit the environment.

5.2.2.2.4.1 Environment of Create and Edit of Professor Dashboard

Professor can interact with the object of the environment.

####### *5.2.2.2.4.1.1 Upload lab environment image*

Professor will need to upload an image of the lab or classroom.

####### *5.2.2.2.4.1.2 Assign Student the Lab of Professor Dashboard*

Professor can give student access and assign a lab to students.

####### *5.2.2.2.4.1.3 Insert Equipments of Professor Dashboard*

Professors can insert one or more lab equipment that were uploaded in ***5.2.2.2.2*** into a lab environment.

####### *5.2.2.2.4.1.4 Save Environment as a Template*

Professors are able to save an already created environment as a

template for future use.

5.2.2.3 Attend Live Lab of Student Dashboard and Staff Dashboard of User Domain

Students and professors are able to attend an active lab with one another.

5.2.2.3.1 Peer Learning of Attend Live Lab of User Domain

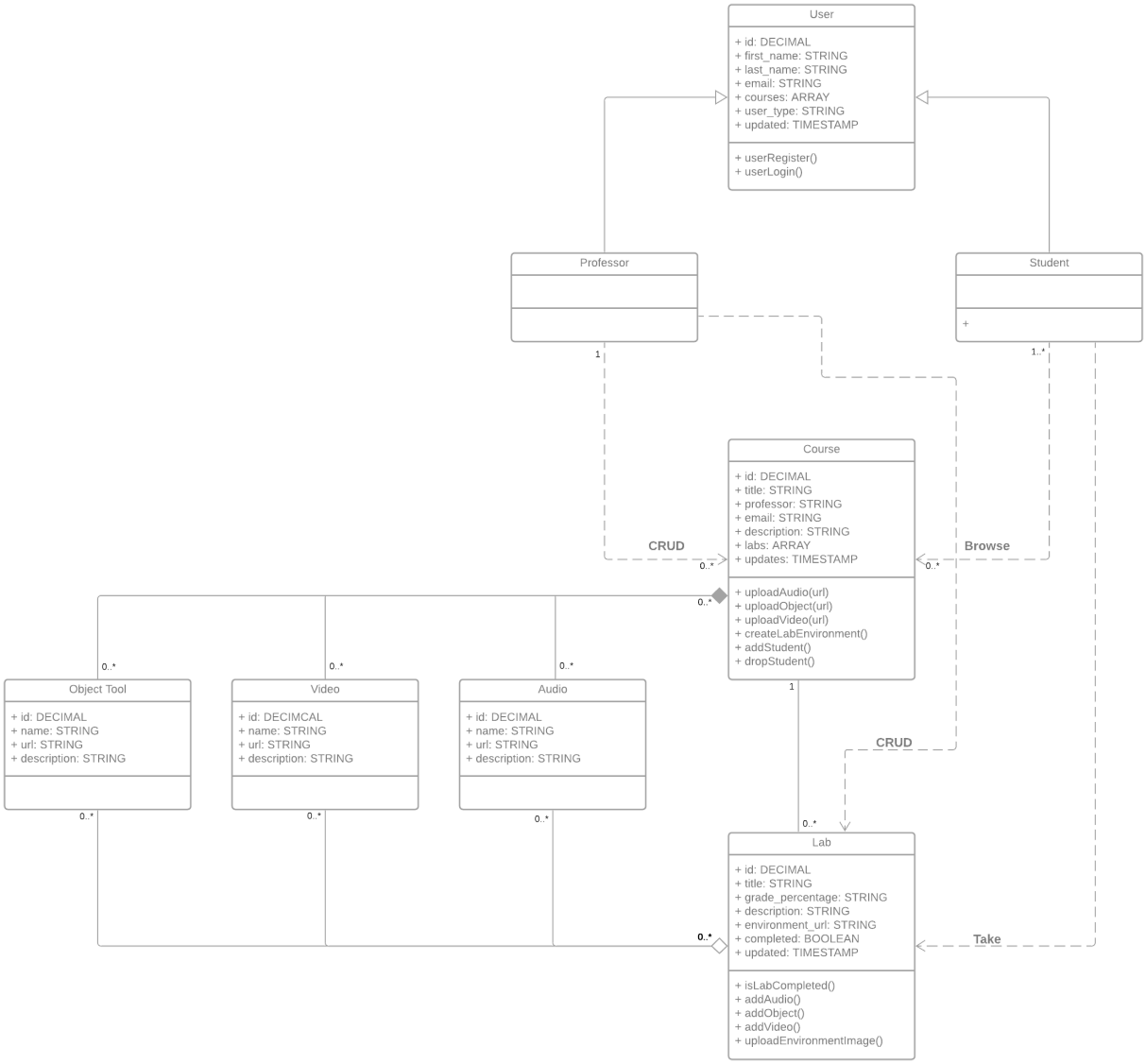
Users can collaborate with peers.

5.2.2.3.2 Interaction of Attend Live Lab of User Domain

Users can interact with tools and equipment.

Section 6 – Data Design

Describe the data contained in databases and other shared structures between domains or within the scope of the overall project architecture



6.1 Persistent/Static Data

Describe/illustrate the logical data model or entity relationship diagrams for the persistent data (or static data if static)

6.1.1 Dataset

Describe persisted object/dataset and its relationships to other entities/datasets

6.1.2 Static Data

Describe static data

6.1.3 Persisted data

Describe persisted data

6.2 Transient/Dynamic Data

Describe any transient data, include any necessary subsections

6.3 External Interface Data

Any external interfaces' data goes here (this is for the data, section 8 is for the interface itself)

6.4 Transformation of Data

Describe any data transformation that goes on between design elements

Section 7 - User Interface Design

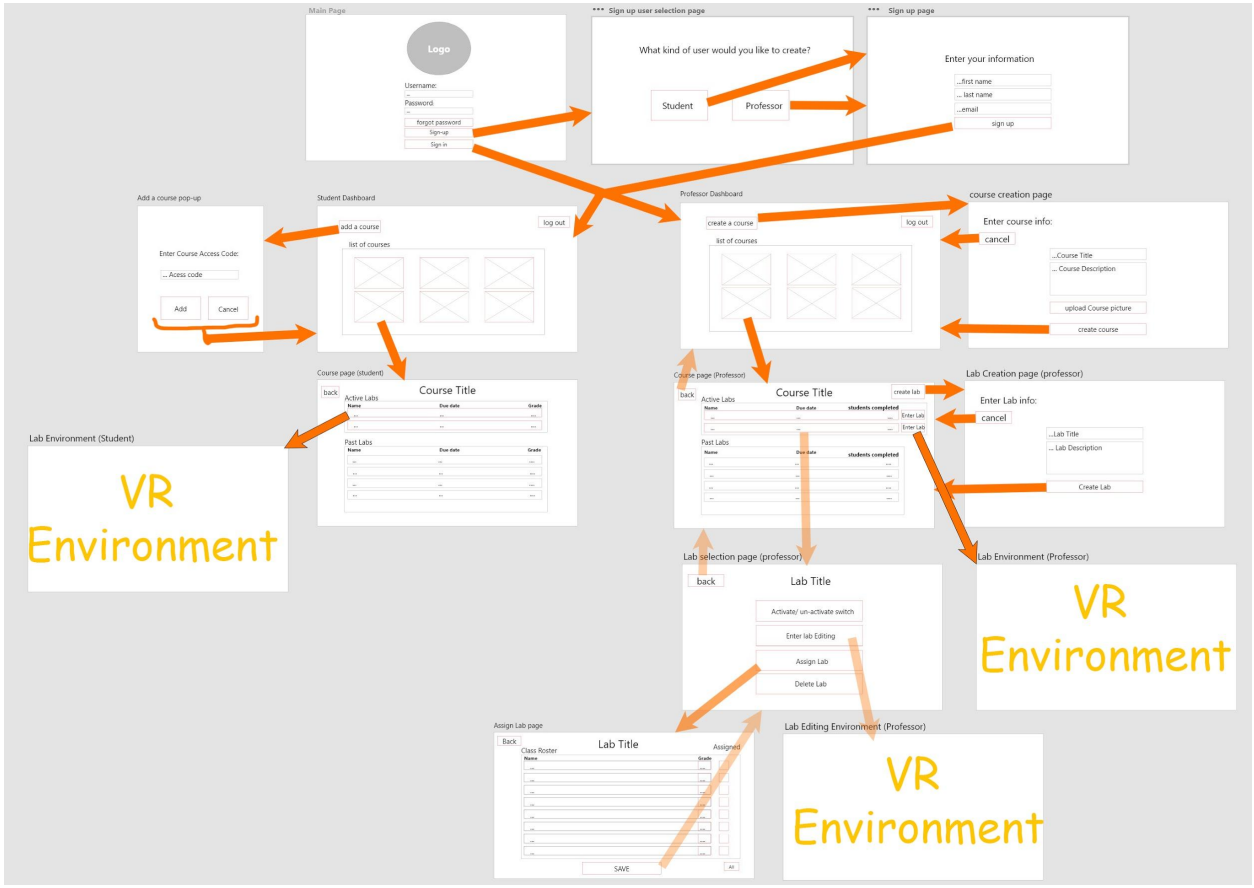
7.1 User Interface Design Overview

Pictures, high level requirements, mockups, etc.

Original size: https://drive.google.com/file/d/1J1JtDyfg6AY3mmqSIdLvj1b4jpty52Jz/view?usp=sharing

7.2 User Interface Navigation Flow

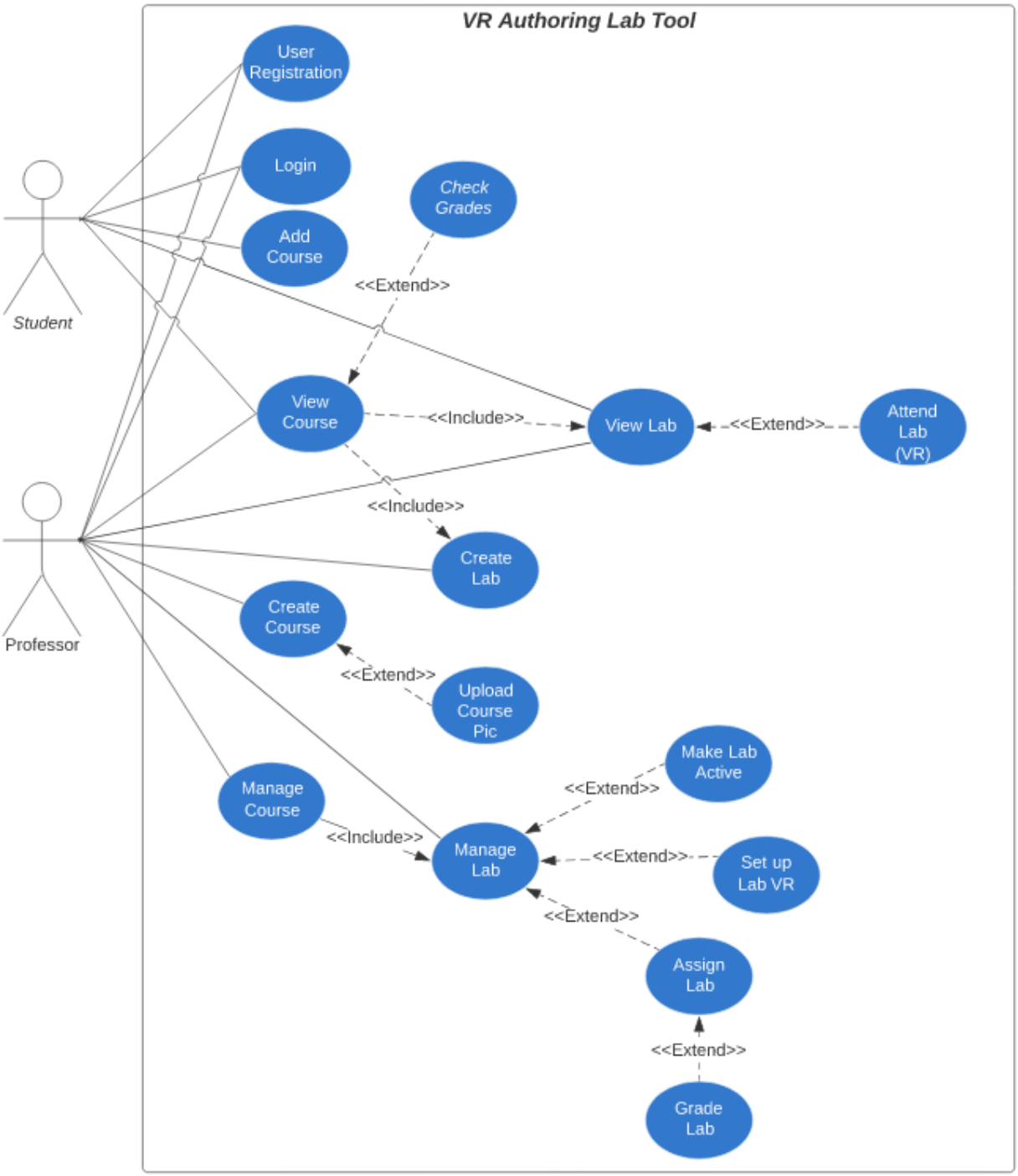
Diagram the flow from one screen to the next



Original size :
https://drive.google.com/file/d/1fRSqDhNBEvigieTWaWQWcqpfEgM9JmJL/view?usp=sharing

7.3 Use Cases / User Function Description

Describe screen usage / function using use cases, or on a per function basis



Section 8 - Other Interfaces

Identify any external interfaces used in the execution of this module, include technology and other

pertinent data

8.1 Interface X

Describe interactions, protocols, message formats, failure conditions, handshaking, etc

Creating a new Student / Professor account:

```
const mongoose = require("mongoose");

const registerSchema = new mongoose.Schema({

   email: {
       type: String,
       required: true,
       unique: true
   },
   password: {
       type: String,
       required: true
   },
   confirmpassword: {
       type: String,
       required: true
   },
   accountType: {
       type: String,
       required: true
   },
   firstName: {
       type: String,
       required: true,
       unique: true
   }
});

//creating a collection
const Register = new mongoose.model("Register", registerSchema);
module.exports = Register;
```

Logging into the application:

```
router.get('/', function(req, res, next) {
```

```
  res.render('login', { title: 'Express' });

});


router.post("/", async(req, res) => {
    try {
         var email = req.body.email;
         var password = req.body.password;

         await User.findOne({email: email,password: password},
function(error, user){
          if(error){
             console.log(error);
             return error;
          }

          //If user is not found
          if(!user) {
              res.send("Incorrect user or password");
             res.redirect("/");
             return error;
          }else { //User exists, so get data and send to dashboard
             var ciphertext = urlSafeEncrypt.enc(JSON.stringify(user));
             res.redirect(`/dashboard?param=${ciphertext}`);

          }
         })

    }
    catch(error) {
      res.status(400).send(error);
    }

});

module.exports = router;
```

Dashboard:

```
router.get('/', async function(req, res, next) {
   var parameter = req.query.param;
```

```
//Get encrypted data
   var decryptedData = await JSON.parse(urlSafeEncrypt.dec(parameter));

   await Labs.find(function(error, lab) {
       if(error) {
           console.log(error);
           return error;
       }
       if(lab) {
           var data = {firstName: decryptedData.firstName, accountType:
decryptedData.accountType, labs: lab, layout: 'dashboard'};
           res.cookie("userID", (decryptedData._id).toString(), {maxAge:
new Date(86400+Date.now())});
           res.render("dashboard", data);
       }
   }).lean();

})


router.get('/lab/:labUrl', async function(req, res, next) {
   try {
       await Labs.findOne({'labUrl': req.params.labUrl}, function(err, lab)
{
           if(err) {
               console.log(err);
               return err;
           }
           if(lab) {
               var data = {
                   fileContent: lab.fileContent,   // append returned
fileContent from database to key
                   layout: 'aframe'     // use aframe_layout.hbs as layout
               };
               res.render('lab', data);
                         }
       });
   }
   catch(err) {
       res.send(err);
   }
});
```

Edit Environment(Professor):

```
fetch('http://localhost:3600/vrenv', {
      Method: 'POST'
      Headers: {
            "Content-type" : "application/json"
            },
      Body: JSON.stringify({
            Object: [
                  Id:decimal
                  name:string
                  Url:string
                  Description:string
            ]
            Environment_url:string
       })
}
```

View Grade:

```
const viewGrade = (title, user_type, id) = {
      id.userGrade();
};

app.get("/grade", viewGrade);
```

View list of Labs for user (Student):

```
const viewLabs = (user_type, id) = {
      id.listLabs();
};

app.get("/labs", viewLabs);
```

Start Lab for user (Student):

```
const startLab = (user_type, id) = {
      id.openLab();
};
app.get("/currentLab", startLab);
```

user(Professor) creating a Course:

```
const coursesSchema = new Schema({
id: DECIMAL
title: STRING
description: STRING
labs: ARRAY
updated: TIMESTAMP
});

const coursesModel = database.model('Courses', courseSchema);

app.post('/courses', [
   Courses.insert
]);
```

user(Professor) creating a Lab:

```
router.post('/save-lab', async (req, res, next) => {
    var title = req.body.title;
    var time = req.body.startTime;
    var date = req.body.startDate;
    var vrScene = req.body.vrScene;
    var uploadedObjFiles = req.body.uploadedObjFiles;
    let buff = new Buffer(uploadedObjFiles, 'base64');
    uploadedObjFiles = buff.toString('ascii');

    try {
        var labUrlString = cryptoRandomString({length: 25, type:
'url-safe'});
        console.log('url-string generated: ' + labUrlString);
        var lab = new Labs({
            labUrl: labUrlString,
            labTitle: title,
```

```
            labTime: time,
            labDate: date,
            labVRScene: vrScene,
            labFileContent: `
                <a-scene>
                    <!-- Asset Management System -->
                    <a-assets>
                      <!-- <img id="celing" src="assets/celing.jpg">   -->
                      <a-asset-item id="lab"
src="assets/laboratory.glb"></a-asset-item>
                      <a-asset-item id="heart-o"
src="assets/corona.obj"></a-asset-item>
                      <a-asset-item id="heart-m"
src="assets/corona.mtl"></a-asset-item>
                      <a-asset-item id="physics-o"
src="assets/physics.obj"></a-asset-item>
                      <a-asset-item id="physics-m"
src="assets/physics.mtl"></a-asset-item>
                      <a-asset-item id="skull-o"
src="assets/skull.obj"></a-asset-item>
                      <a-asset-item id="skull-m"
src="assets/skull.mtl"></a-asset-item>
                    </a-assets>
                    <!-- Celing -->
                    <!-- <a-sky src="#celing"></a-sky> -->

                    <!-- Lab model -->
                    <a-entity gltf-model="#lab"></a-entity>

                    <!-- Sun and Sky model -->
                    <a-simple-sun-sky sun-position="0.5 0.5
1"></a-simple-sun-sky>
                    <a-entity light="type: ambient; color:
#BBB"></a-entity>
                    <a-entity id="directional" light="type: directional;
color: #FFF; intensity: 1.3" position="0.5 1 2"></a-entity>
                    <a-entity id="directional" light="type: directional;
color: #FFF; intensity: 1.3" position="-0.5 1 -2"></a-entity>

                    <!-- Heart model -->
                     <a-entity obj-model="obj: #heart-o; mtl: #heart-m"
                                        position="-0.5 1 0.5"
```

```
                                        scale="0.001 0.001
0.001"></a-entity>
                    <a-entity obj-model="obj: #physics-o; mtl: #physics-m"
                                        position="-0.1 1.2 0.5"
                                        scale="0.002 0.002
0.002"></a-entity>
                    <a-entity obj-model="obj: #skull-o; mtl: #skull-m"
                                        position="2 1.2 0.5"
                                        scale="20 20 20"></a-entity>

                </a-scene>`

        });


        await lab.save();
        // if lab was saved successfully, return 200 and a message
        res.status(200).send({message: title + 'Lab saved.'});
    } catch(err) {
        // error occurred during saving, return 500 and a message
        res.status(500).send({name: err.name, message: err.message});
    }

});
```

user(Professor) uploading a video:

```
const uploadSchema = new Schema({
id: DECIMAL
name: String
url: String
description: String
});

const uploadVideoModel = database.model('uploadVideo', uploadSchema);

app.post('/video', [
   uploadVideo();
]);
```

user(Professor) uploading audio:

```
const uploadSchema = new Schema({
id: DECIMAL
name: String
url: String
description: String
});

const uploadAudioModel = database.model('uploadAudio', uploadSchema);

app.post('/audio', [
   uploadAudio();
]);
```

Uploading Object Tools:

```
const uploadSchema = new Schema({
id: DECIMAL
name: String
url: String
description: String
});

const uploadObjectModel = database.model('uploadObject', uploadSchema);

app.post('/object', [
   uploadObject();
]);
```

Save Class Environment Template:

```
Const saveClassTempSchema = new schema ({
      environment_url: String
      labtitle:String
      description:String
)};

Const saveClassTempModel = database.model('saveClassTemp',
saveClassTempSchema);
```

```
app.post('/classTemp', [
      classTempController.save
]);
```

Section 9 - Extra Design Features / Outstanding Issues

Does not fit anywhere else above, but should be mentioned -- goes here

Section 10 – References

Any documents which would be useful to understand this design document or which were used in drawing up this design.

Section 11 – Glossary

Glossary of terms / acronyms