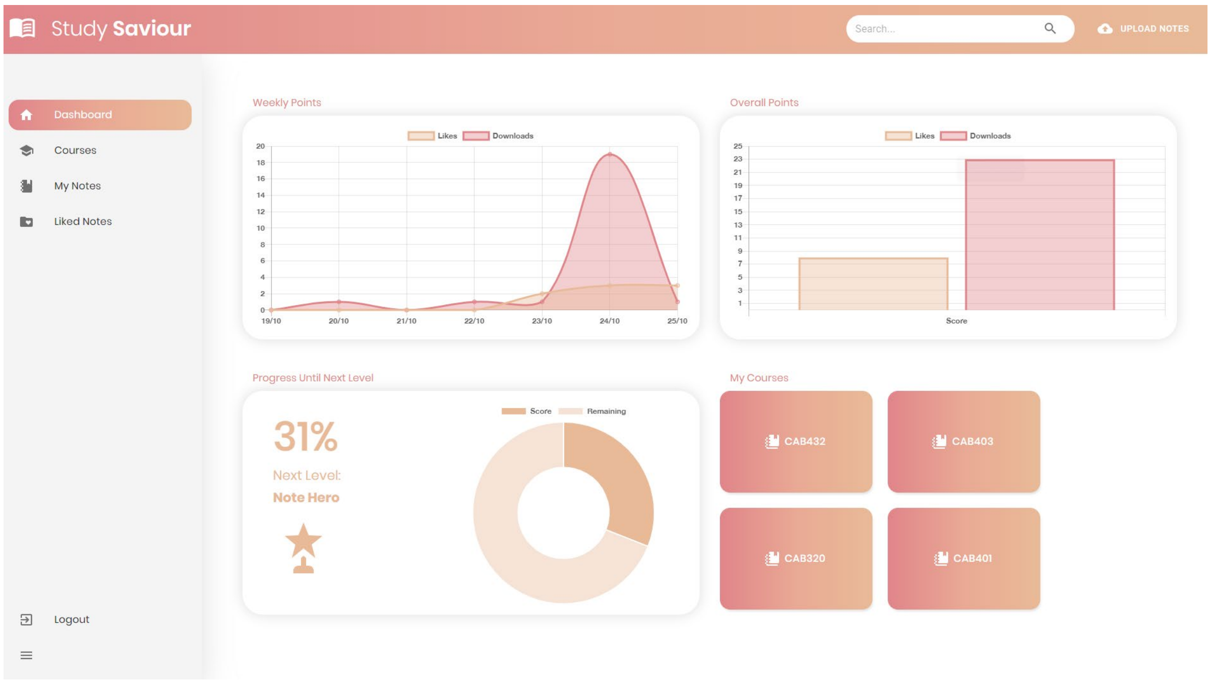
Study **Saviour**



Liam Percy N9959807 25/10/2020

Contents

Introduction

Purpose & Description

Collaboration is an important part of the university experience. A great way to further ones understanding of a topic in a particular unit, or enhance learning in general, is to join a study group.

These groups provide students with a means for sharing and comparing class notes to better understand important concepts that they may have missed during lectures or tutorials; sharing knowledge to gain unique insights into a particular group members understanding of a topic; covering more material by assigning topics to individual group members to research; as well as a whole host of other benefits.

Despite this, many students do not engage in, or have access to one of these groups. This impedes their opportunity for growth, and generally means they have a lesser understanding of concepts, compared to those students who do engage in study groups.

Study Saviour is a solution to this problem, providing students with a platform to share notes with each other. Students can join the platform, and then start uploading notes for a particular course. They can also enrol in courses themselves to see notes that have been uploaded by other students.

Notes can also be liked by other students, which in turn saves the note to the student's account. This provide them with the ability to store information relevant to them, but more importantly, it provides the platform with a key metric for sorting notes to be displayed to users. The most liked (and generally the highest quality) notes will be displayed first.

Finally, these likes are tracked and displayed on a dashboard to the user. A graph for daily likes and downloads is displayed, and a ranking system is used to gamify the experience and encourage users to regularly upload high quality notes.

Services Used

Firebase Authentication

Provides authentication services using email and password combinations in order to access the application, securely save user data in the cloud, and provide a personalised experience by serving content relevant to the user.

Docs: https://firebase.google.com/docs/auth

Endpoint:

```
firebase.auth().signInWithEmailAndPassword(email, password); // Sign in
```

Firebase Cloud Firestore

A stateless database solution used for storing, synchronising, and retrieving application data, including user information such as enrolled courses, likes and uploaded notes, the notes and courses themselves, and also various tags that any particular note can have.

Docs: https://firebase.google.com/docs/firestore

Endpoint:

```
import admin from "firebase-admin"; // Node.js import
const db = admin.firestore(); // Add database
const userRef = db.collection("users"); // Add document reference
const doc = await userRef.doc(user).get(); // Get document
```

Firebase File Storage

Used for uploading, storing, and retrieving note file objects. Objects are stored in in a Google Cloud Storage bucket as a PDF document.

Docs: https://firebase.google.com/docs/storage

Endpoint:

```
const storageRef = firebase.storage().ref(); // Storage reference
// Get download URL using note ID and note name
storageRef.child(`notes/${noteId}/${noteName}.pdf`).getDownloadURL()
```

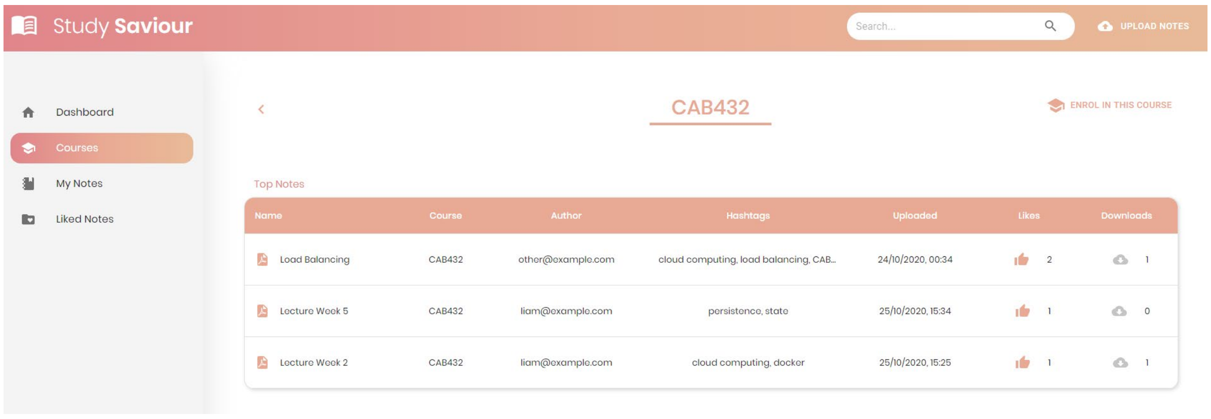
Use Cases

Access to Study Notes

As a student, I want to be able to see high quality study notes for a particular course, so that I can better understand important concepts that I have missed during lectures and tutorials, gain unique insights into another students understanding of a topic, and overall enhance my learning experience.

To achieve this, a user can open the app, and go to the courses page. This will load all of the courses stored in Firestore or cached by Redis. By clicking on the course that is relevant to them, they will see a list of notes uploaded for that course, sorted by the number of likes a particular note has. This increases the likelihood that a user will be exposed to high quality study notes when using the application. This information will also be served using a either Firestore or Redis.

If a user frequently accesses notes from this course, they can *enrol* in the course, which will add it to their list of courses in the Cloud Firestore. The course will then be visible on the user's dashboard page, under the *my courses* section.



Name	Course	Author	Hashtags	Uploaded	Likes	Downloads
Load Balancing	CAB432	other@example.com	cloud computing, load balancing, CAB...	24/10/2020, 00:34	2	1
Lecture Week 5	CAB432	liam@example.com	persistence, state	25/10/2020, 15:34	1	0
Lecture Week 2	CAB432	liam@example.com	cloud computing, docker	25/10/2020, 15:25	1	1

Figure 1: Notes

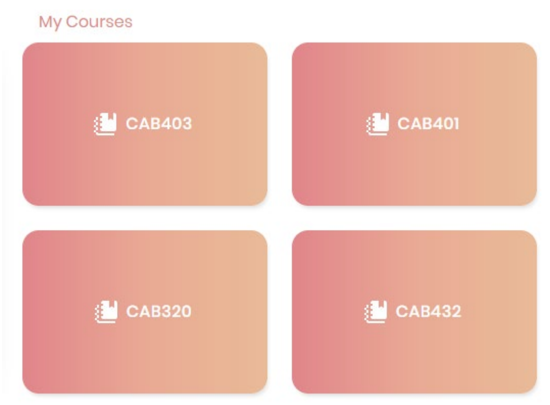


Figure 2: My Courses

Upload Gamification

As a frequent note uploader, I want a ranking system based on my upload likes and download statistics, so that I can feel appreciated for my contributions to the platform. I would also like to track these statistics, so that I can monitor my progress towards my next rank and feel encouraged to continue providing documents to the service.

To achieve this, a user simply needs to log into the application and view their dashboard page, which is the default page loaded for any given session. This page displays the past weeks statistics on likes and downloads for the user's notes, in addition to their overall downloads and likes.

Additionally, a third graph shows the users progress towards their next rank, based on their current score. Score is a metric based on the total number of likes and downloads a user has for their notes, and each rank is specified by a set number in the codebase.

This information is retrieved for Firestore or Redis if it has been cached. If another user likes one of the users notes, then the cached results are removed, and the new results are loaded. This ensures that a user's information is always up to date and improves network performance if the information has not changed.

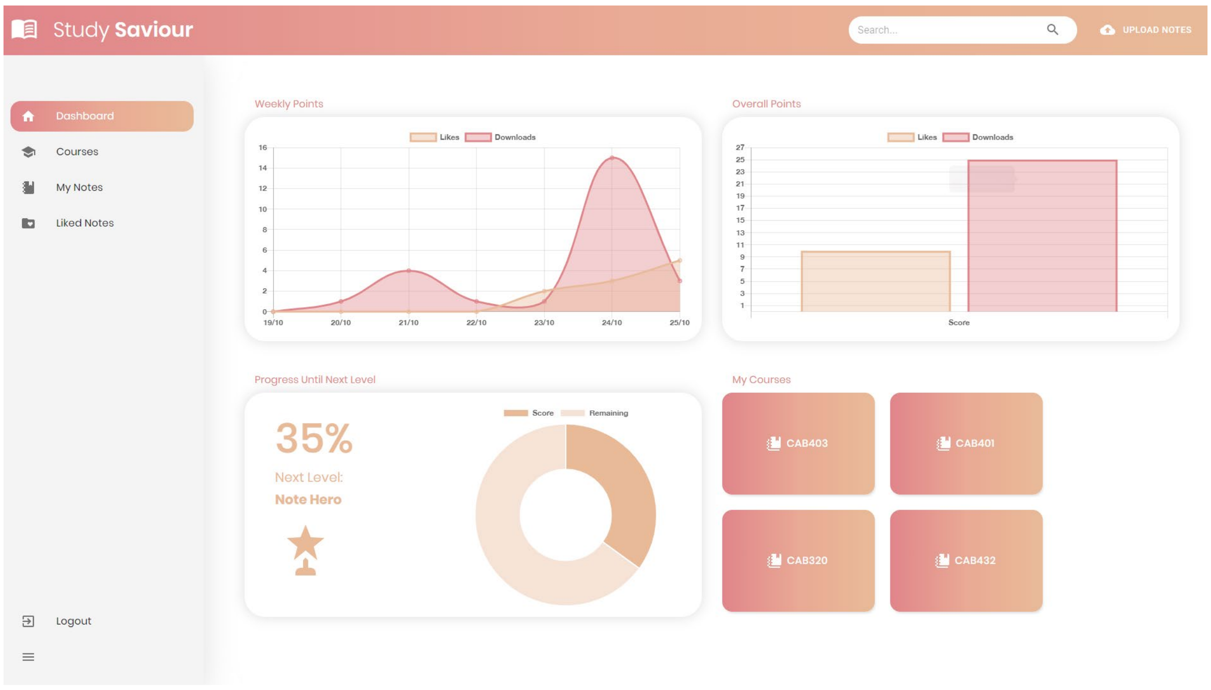


Figure 3: Dashboard

Technical Breakdown

Architecture

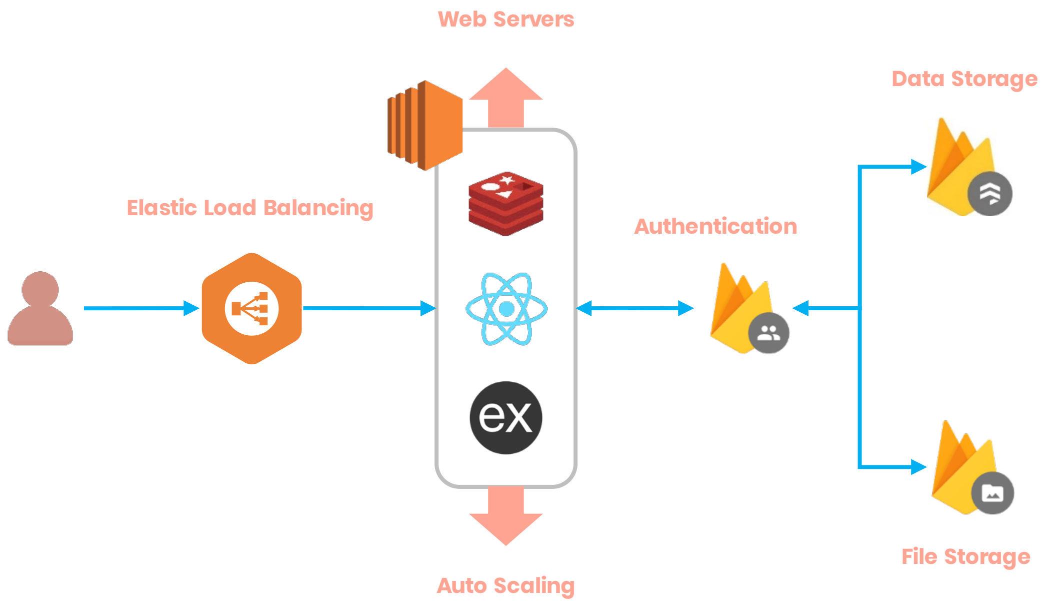
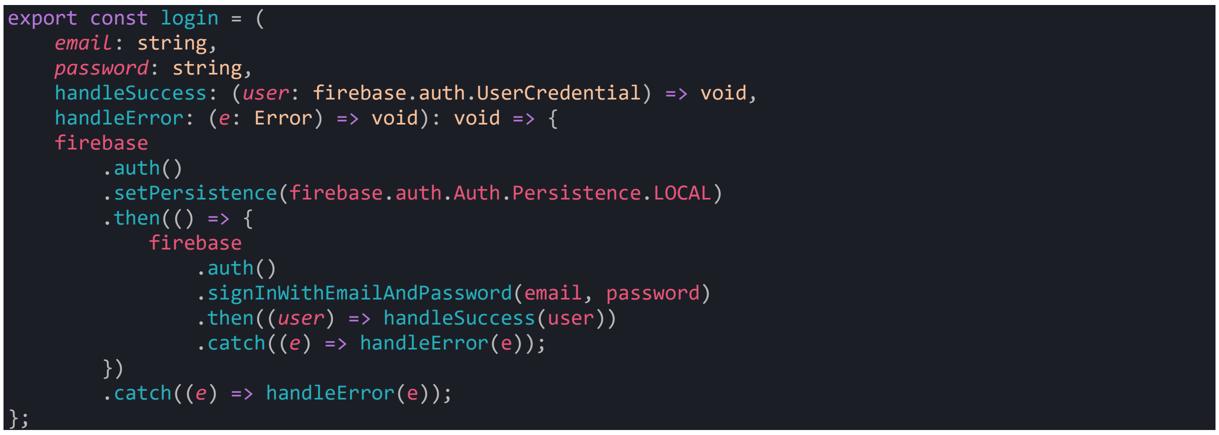


Figure 4: Network Architecture

Using the figure above, we can step through the system architecture to understand how a user interacts with the system. Initially, they will access the application through an elastic application load balancer. This load balancer is configured to route users to EC2 instances in an auto-scaling group, using a round robin technique.

From the EC2 instance, a user will login to the application using Firebase Authentication. Once authenticated, users can read from or write data to Firebase Cloud Firestore, and upload/download files from Firebase Cloud Storage.



```
export const login = (
    email: string,
    password: string,
    handleSuccess: (user: firebase.auth.UserCredential) => void,
    handleError: (e: Error) => void): void => {
    firebase
        .auth()
        .setPersistence(firebase.auth.Auth.Persistence.LOCAL)
        .then(() => {
            firebase
                .auth()
                .signInWithEmailAndPassword(email, password)
                .then((user) => handleSuccess(user))
                .catch((e) => handleError(e));
        })
        .catch((e) => handleError(e));
};
```

Figure 5: Login Function

```
export const logout = (
    handleSuccess: () => void,
    handleError: (e: Error) => void): void => {
    firebase
        .auth()
        .signOut()
        .then(() => {
            handleSuccess();
        })
        .catch((e) => {
            handleError(e);
        });
};
```

Figure 6: Logout Function

```
export const upload = (
    file: File,
    noteId: string,
    noteName: string,
    handleSuccess: (val: firebase.storage.UploadTaskSnapshot) => void
) => {
    const storageRef = firebase.storage().ref();
    const noteRef = storageRef.child(`notes/${noteId}/${noteName}.pdf`);
    noteRef.put(file).then((snapshot) => handleSuccess(snapshot));
};
```

Figure 7: Upload Function

These functions are all completely stateless. Login credentials, data and files are all held in Firebase, and this information does not depend on transient state between service invocations. If a single instance fails, a user can simply connect to another; all of their data will still persist externally.

Additionally, the application uses Redis as a form of cached in-memory state. When a user makes a request to a particular endpoint, the server will first check to see if a cached entry exists for the requested data. If it does, Redis will serve the data, as apposed to Firebase. Because every endpoint relies on the logged in user, and keys are frequently deleted due to updates to likes/notes/downloads, this served purely as a performance enhancement technology, and was also used to prevent against duplicate read/write requests to Firebase resulting unnecessary costs.

```
app.post("/api/user", async (req, res) => {
    try {
        const { user } = req.body;
        const redisKey = `/api/user/${user}`;
        redisClient.get(redisKey, async (err, result) => {
            if (result) {
                const user = JSON.parse(result);
                console.log("User from Redis");
                res.status(200).json(user);
            } else {
                const doc = await userRef.doc(user).get(); // Get document
                redisClient.setex(redisKey, 3600, JSON.stringify(doc.data()));
                console.log("User from Firestore");
                res.status(200).json(doc.data());
            }
        });
    } catch (error) {
        console.log(error);
        res.status(500).send(error);
    }
});
```

Figure 8: Using Redis to Get User Data

Responsibilities

Client

The client itself uses the React framework to render HTML, CSS and JavaScript to the user. All authorisation and file upload/download functions (see figures 5, 6, 7, 9) are handled by the client. In saying that, these functions are just function calls to Firebase, which will then return the necessary information. The download function does post information to the server, which then makes another call to Firestore in order to increase the number of downloads for a particular note.

```
const openNote = () => {
        Axios.post("/api/notes/download-note", { note: note.id });
        const downloads = state.downloads + 1;
        setState({ ...state, downloads: downloads });
        const storageRef = firebase.storage().ref();
        storageRef
            .child(`notes/${note.id}/${note.name}.pdf`)
            .getDownloadURL()
            .then((url) => window.open(url));
    };
```

Figure 9: Opening a Note in the Browser

Authorisation is used to prevent users from accessing the application if they are note logged in. React Router is used to render content based on the current page route. A custom Private Route component was used. This component gets the current logged in user and allows them to access the route. If the user is not logged in, then they will be redirected to the login screen.

```
export default function PrivateRoute({ component: Component, ...rest }) {
    const history = useHistory();
    const user = firebase.auth().currentUser;
    return (
        <Route
            {...rest}
            render={(props) => {
                return user ? (
                    <Component {...props} />
                ) : (
                    <Redirect to={`/login?redirect=${history.location.pathname}`} />
                );
            }}
        />
    );
}
```

Figure 10: Private Route

```
export default function Routes() {
    return (
        <Router>
            <Switch>
                <PrivateRoute exact path="/" component={Home} />
                // ... other routes
                <Route path="/login" component={Login} />
                <Route path="/register" component={Register} />
                <Route path="/404" component={PageNotFound} />
                <Redirect to="/404" />
            </Switch>
        </Router>
    );
}
```

Figure 11: Private Route Usage

Server

The server uses the Express.JS framework to make requests to both our Redis Client and Firebase. All requests that involve accessing the Cloud Firestore occur here. The server starts by initialising the Redis Client, and the Firebase application, before it then sets up the Firestore collection references.

```
// Redis client
const redisClient = redis.createClient();

// Firebase application
admin.initializeApp({
    credential: admin.credential.cert(service),
    databaseURL: "https://study-saviour.firebaseio.com",
});

// Firestore noSQL database
const db = admin.firestore();
const userRef = db.collection("users");
const likesRef = db.collection("likes");
const notesRef = db.collection("notes");
const hashtagsRef = db.collection("hashtags");
const coursesRef = db.collection("courses");
```

Figure 12: Server Initialisation

Once this has been completed, the server can make queries to the Redis Cache, or Firestore, or post data. Finally, the server also serves static page content to the client.

```
// Define build path
const __dirname = path.dirname(url.fileURLToPath(import.meta.url));

// Serve static assets built from the clientside
app.use(express.static(path.join(__dirname, "../client/build")));

// All other routes will be served clientside content
app.use((req, res) => {
    res.sendFile(path.join(__dirname, "../client/build", "index.html"));
});
```

Figure 13: Server Client-side Content

Data Manipulation

The Firestore data was structured in such a way as to prevent significant server-side processing, however one helper function was developed in order to append whether a user has liked a particular note or not. This is then used in the frontend to determine how a like is rendered (coloured or not), and the action we should take (like, unlike). See Appendix 12/13 for more detail

```
function appendLikes(sortedNotes, userLikes) {
    sortedNotes.forEach((note) => {
        if (userLikes.includes(note.id)) {
            note.liked = true;
        } else {
            note.liked = false;
        }
    });
    return sortedNotes;
}
```

Figure 14: Append User Liked Note

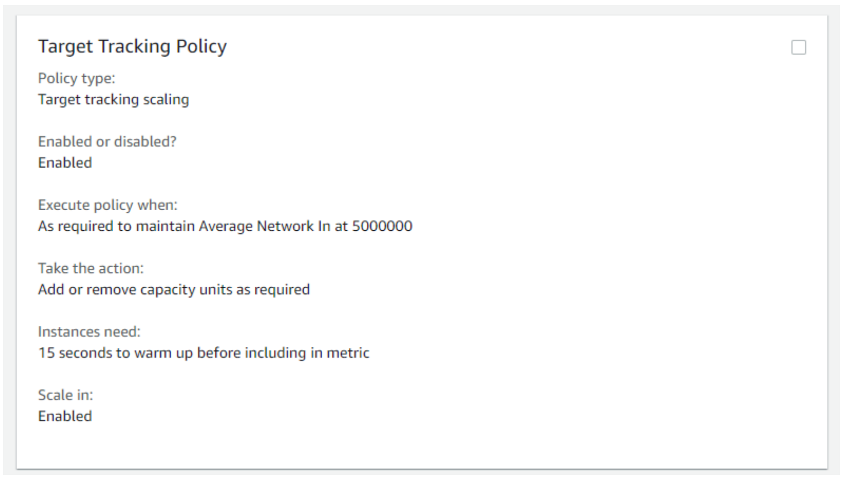
Scaling Performance

When considering which metric to use when scaling an application, it is important to consider the applications use-case. What metric is the most important under high-stress loads? In the case of Study Saviour, one real world example of potential load increase would be during the exam period at university.

As students begin to cram for exams at the end of the semester, many of them may download a significant number of notes and documents in order to aid them with their studies. As such, the application network activity will increase significantly, and a single instance may not be able to handle such high demand.

Referring back to our network architecture diagram in figure 4, we can observe that we receive requests from both users, looking for data/files, and Google, who take these requests, gather the required information, and send back a response to the server, which then displays this information to the client. As such, the decision was made to setup an application scaling policy based on the average number of bytes received out on all network interfaces by the Auto Scaling Group.

An estimation was made that for any given period of significant load on the server, a new request would come in approximately every 5 seconds. Based on initial testing, this resulted in peaks of 10mB of data transfer every 5 minutes, for small sized PDFs documents. Based on this information, a simple policy was created for incoming network activity at 5mB.


Target Tracking Policy

Policy type:
Target tracking scaling

Enabled or disabled?
Enabled

Execute policy when:
As required to maintain Average Network In at 5000000

Take the action:
Add or remove capacity units as required

Instances need:
15 seconds to warm up before including in metric

Scale in:
Enabled

Figure 15: Target Tracking Policy

Testing this policy, we can see that as we receive large amounts of network data, our application scales out in order to better handle the network load. Once these requests subside, the application scales back in, to ensure we are not using more instances than we require.

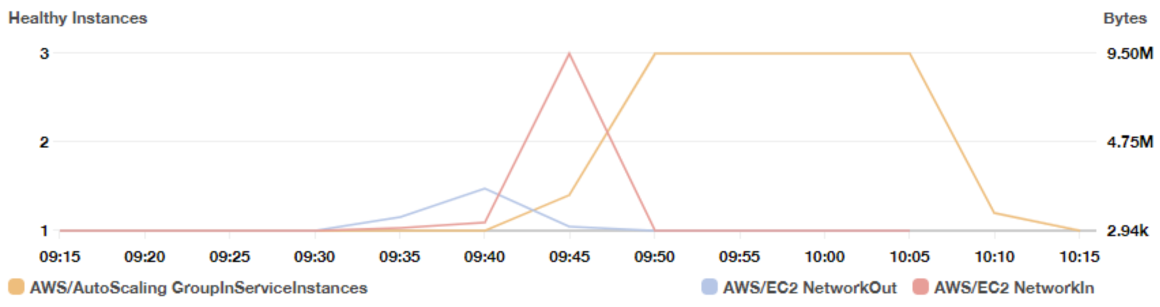
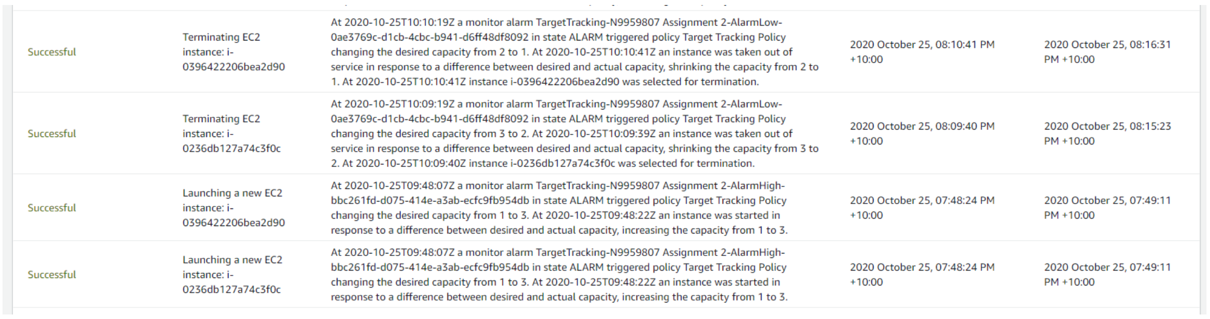


Figure 15: Network Usage and Instance Count Scaling

Additionally, we can observe the scaling pool instance creation and destruction events, based on this policy.



Successful	Terminating EC2 instance: i-0396422206bea2d90	At 2020-10-25T10:10:19Z a monitor alarm TargetTracking-N9959807 Assignment 2-AlarmLow-0ae3769c-d1cb-4cbc-b941-d6ff48df8092 in state ALARM triggered policy Target Tracking Policy changing the desired capacity from 2 to 1. At 2020-10-25T10:10:41Z an instance was taken out of service in response to a difference between desired and actual capacity, shrinking the capacity from 2 to 1. At 2020-10-25T10:10:41Z instance i-0396422206bea2d90 was selected for termination.	2020 October 25, 08:10:41 PM +10:00	2020 October 25, 08:16:31 PM +10:00
Successful	Terminating EC2 instance: i-0236db127a74c3f0c	At 2020-10-25T10:09:19Z a monitor alarm TargetTracking-N9959807 Assignment 2-AlarmLow-0ae3769c-d1cb-4cbc-b941-d6ff48df8092 in state ALARM triggered policy Target Tracking Policy changing the desired capacity from 3 to 2. At 2020-10-25T10:09:39Z an instance was taken out of service in response to a difference between desired and actual capacity, shrinking the capacity from 3 to 2. At 2020-10-25T10:09:40Z instance i-0236db127a74c3f0c was selected for termination.	2020 October 25, 08:09:40 PM +10:00	2020 October 25, 08:15:23 PM +10:00
Successful	Launching a new EC2 instance: i-0396422206bea2d90	At 2020-10-25T09:48:07Z a monitor alarm TargetTracking-N9959807 Assignment 2-AlarmHigh-bbc261fd-d075-414e-a3ab-ecfc9fb954db in state ALARM triggered policy Target Tracking Policy changing the desired capacity from 1 to 3. At 2020-10-25T09:48:22Z an instance was started in response to a difference between desired and actual capacity, increasing the capacity from 1 to 3.	2020 October 25, 07:48:24 PM +10:00	2020 October 25, 07:49:11 PM +10:00
Successful	Launching a new EC2 instance: i-0236db127a74c3f0c	At 2020-10-25T09:48:07Z a monitor alarm TargetTracking-N9959807 Assignment 2-AlarmHigh-bbc261fd-d075-414e-a3ab-ecfc9fb954db in state ALARM triggered policy Target Tracking Policy changing the desired capacity from 1 to 3. At 2020-10-25T09:48:22Z an instance was started in response to a difference between desired and actual capacity, increasing the capacity from 1 to 3.	2020 October 25, 07:48:24 PM +10:00	2020 October 25, 07:49:11 PM +10:00

Test Plan

Task	Expected Outcome	Result	Appendix
Login	Users can login with authenticated account.	PASS	1
Handle Login Error	Error messages are provided to the user if their account does not exist, or their input is invalid		2
Logout	Users can logout by clicking the logout button	PASS	3
Create Account	Users can create a new account using a username and password.	PASS	4
Handle Create Account Error	Error messages are provided to the user if their input is invalid		5
Load Dashboard	Dashboard displays a timeline of likes and downloads for a user's notes, overall stats, progress towards their next rank and their currently enrolled courses	PASS	6
Load Courses	All available courses displayed	PASS	7
Load Course	List of notes posted for the selected course displayed, sorted by likes	PASS	8
Load My Notes	List of notes uploaded by the user displayed	PASS	9
Load Liked Notes	List of the users liked notes displayed	PASS	10
Load All Cached	All of the above load functions are cached, and can be loaded again using Redis	PASS	11
Like Note	The notes like counter increases by one, the like button changes colour, if the page is refreshed, sorting changes, and the author of the note sees the statistic on their dashboard	PASS	12
Unlike Note	The notes like counter decreases by one, the like button changes colour, if the page is refreshed, sorting changes, and the author of the note sees the statistic on their dashboard	PASS	13
Download Note	The downloaded note is displayed to the user in the browser	PASS	14
Access Unauthorised Route	The user will be directed to the login page. Upon successfully logging in, they will be directed to the page they originally tried to access.	PASS	15
Upload Note	Note uploaded with the correct information	PASS	16
Handle Upload Note Errors	Error messages displayed for invalid requests	PASS	17
Scale Out	The application increases its instance pool based on the network load	PASS	-
Scale In	The application decreases its instance pool based on the lack of network load	PASS	-

Exclusions & Extensions

Overall, the scope for this particular project was quite large. You may have noticed a search bar in the top of the header that does nothing. There are many more features I could develop for the application that would take months to complete. These include searching, sorting results by week/month/all time, filtering results for when the dataset gets too large, user account customisation, stateless Redis caching and more.

While I will most likely continue development of this application, these additions were outside of the scope of this assignment, and I felt at though the application was very feature rich in its current state already.

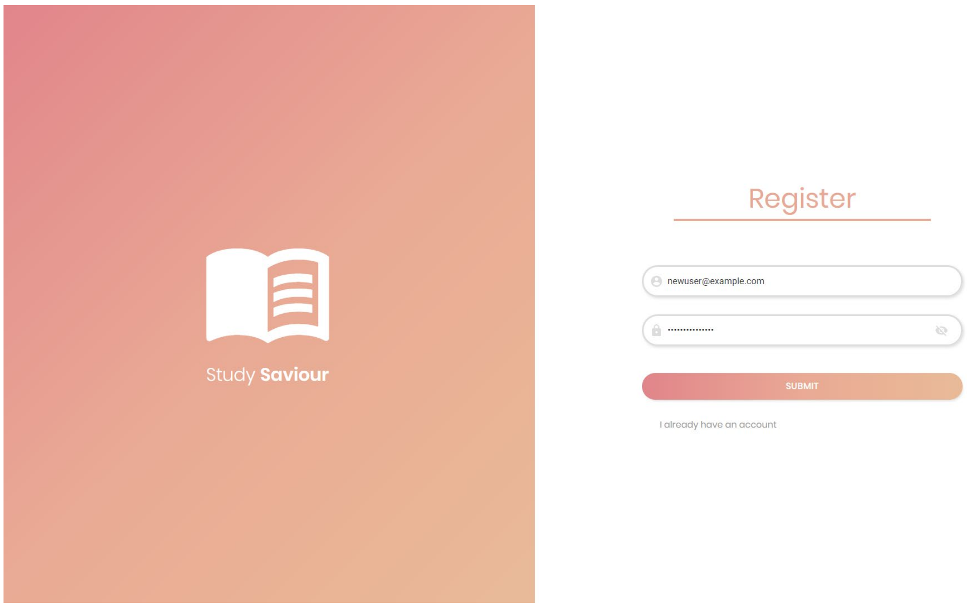
User Guide

Create an Account

You can create an account from the login screen by selected the create account link

Create an account

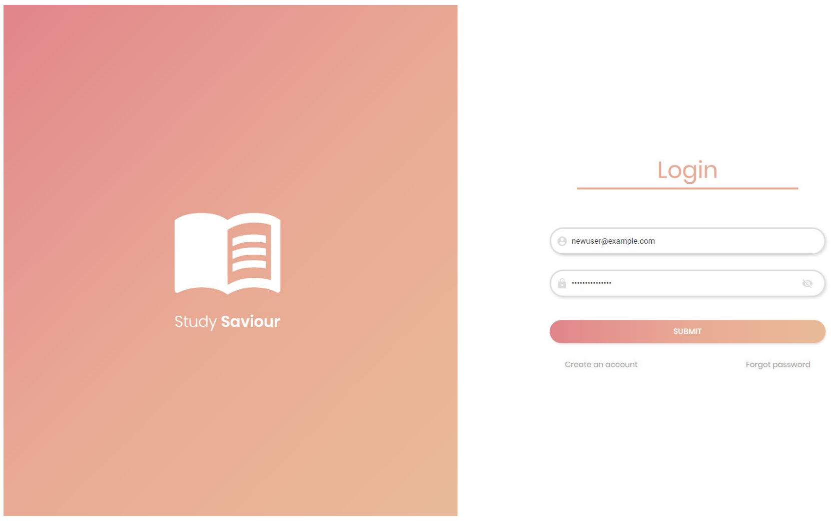
This will take you to the registration page. Enter your desired username and password, and the click submit.



newuser@example.com
•••••••••••••••

Login

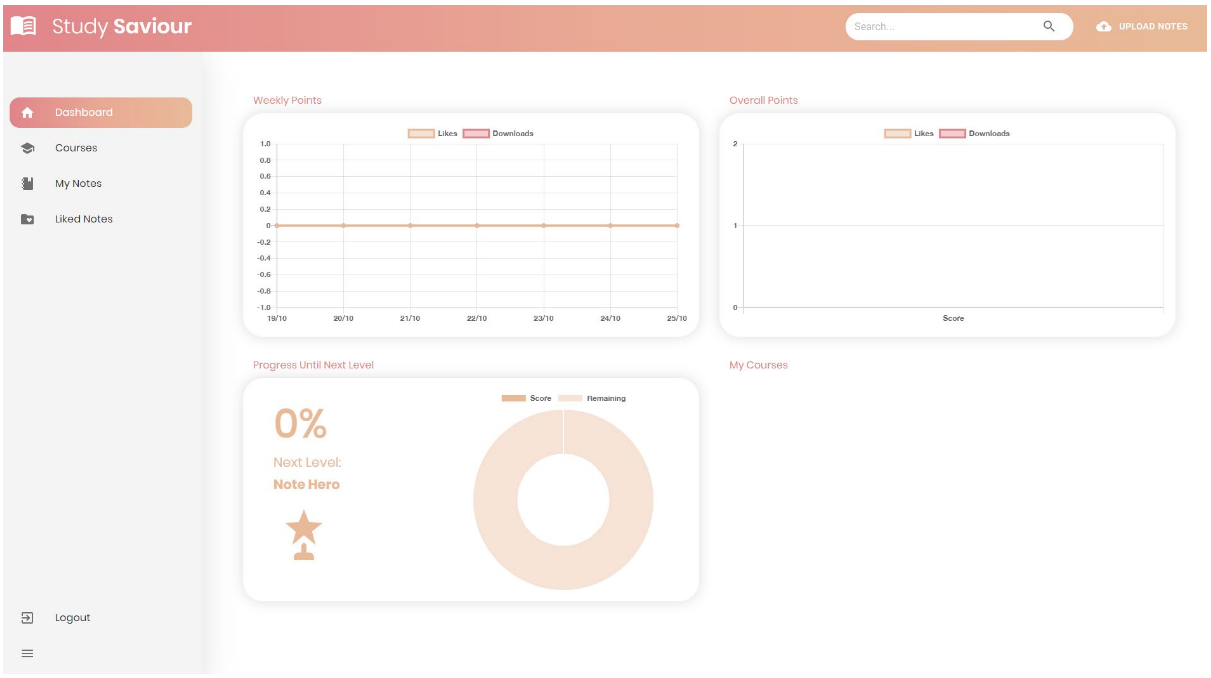
Once you have created an account, you can user these credentials to login to the application



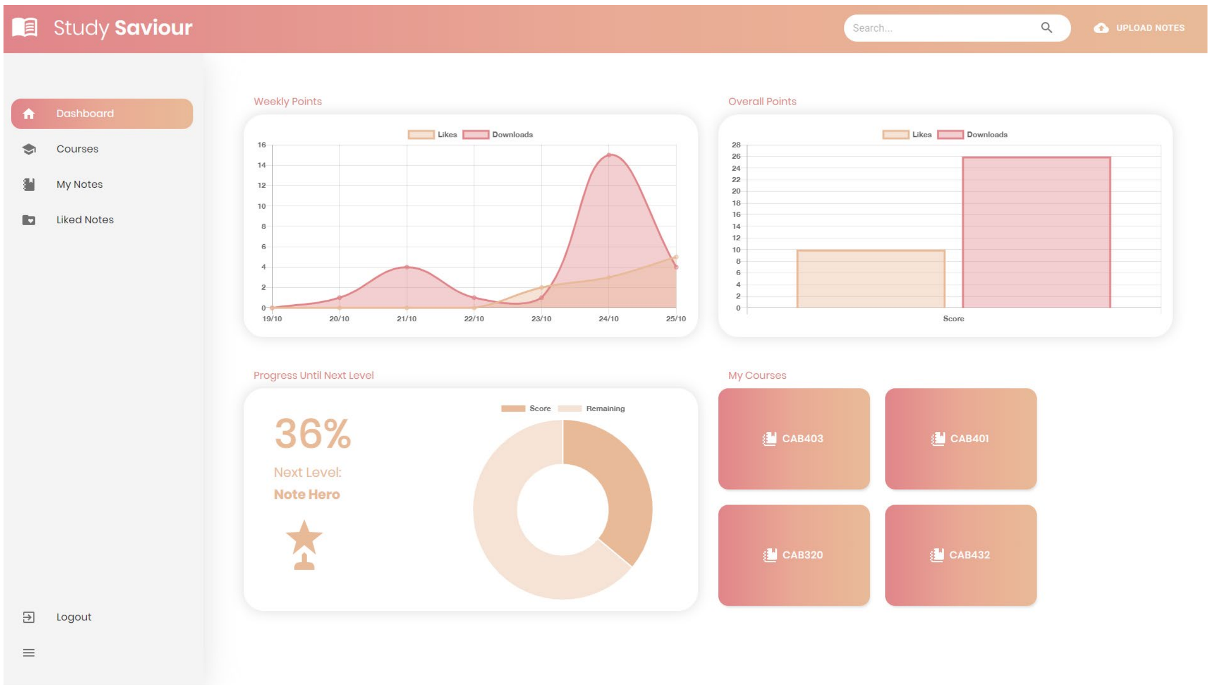
newuser@example.com
•••••••••••••••

View Dashboard

Once logged in, you will see your dashboard, here you will see statistics for your notes including likes and downloads, and also your progress towards ranking up as a note taker!

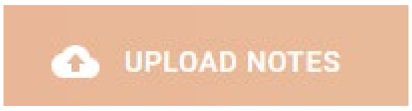


As you upload notes and continue using the system, you will make progress, and your dashboard will evolve.

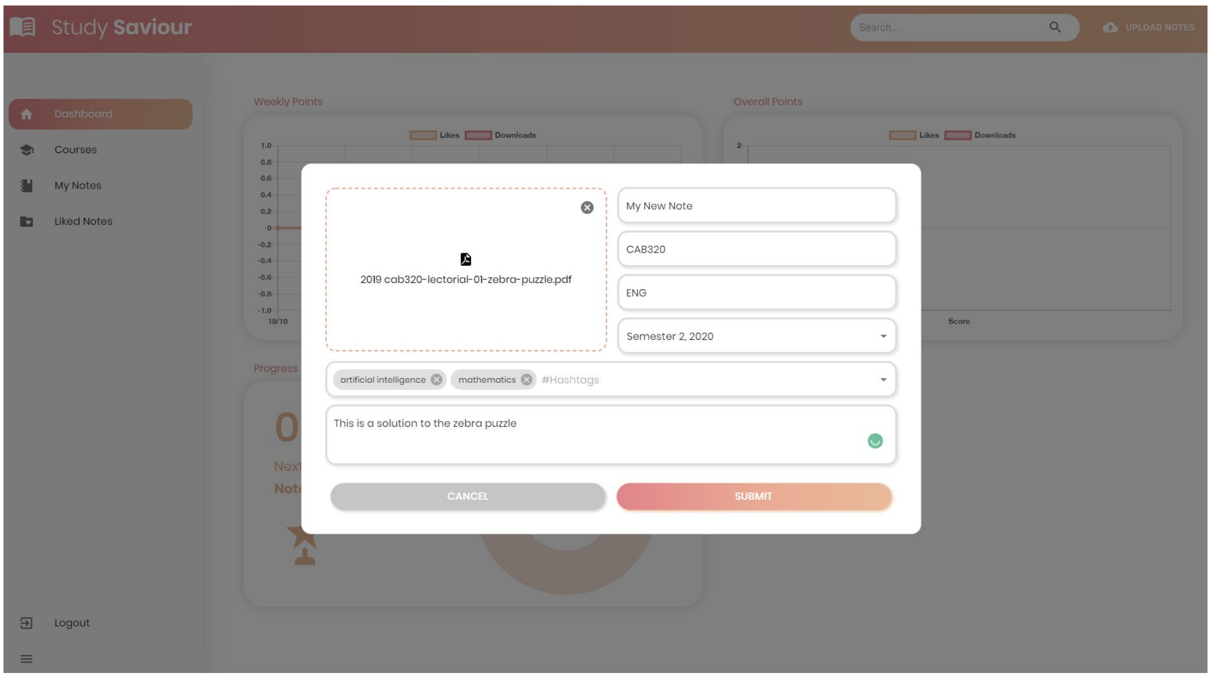


Upload Note

So lets upload our first note. Select the upload notes button at the top of the application.



Enter information into all of the fields, and once you have finished, click submit



2019 cab320-lectorial-01-zebra-puzzle.pdf

My New Note

CAB320

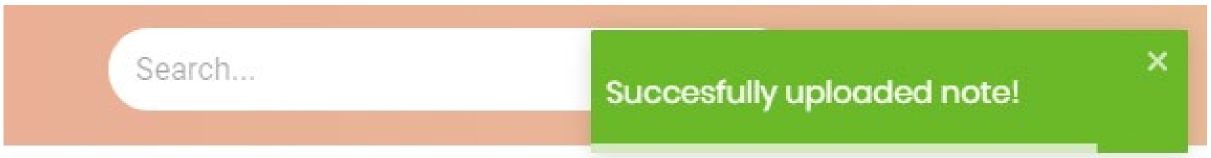
ENG

Semester 2, 2020

artificial intelligence, mathematics, #Hashtags

This is a solution to the zebra puzzle

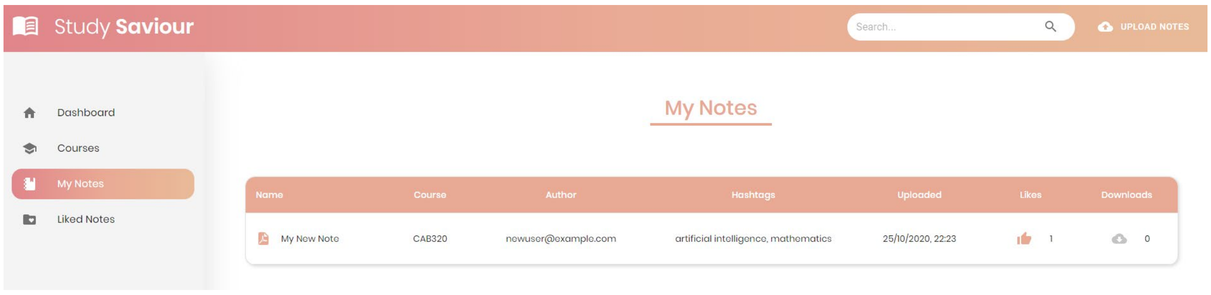
You will see a success notification appear in the top left of the screen



Succesfully uploaded note!

View Notes

Once you have uploaded your note, you can view it by clicking the my notes button in the navigation bar.

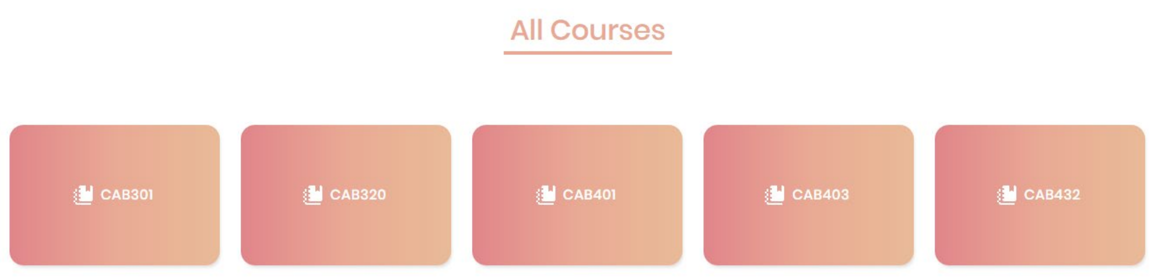


My Notes

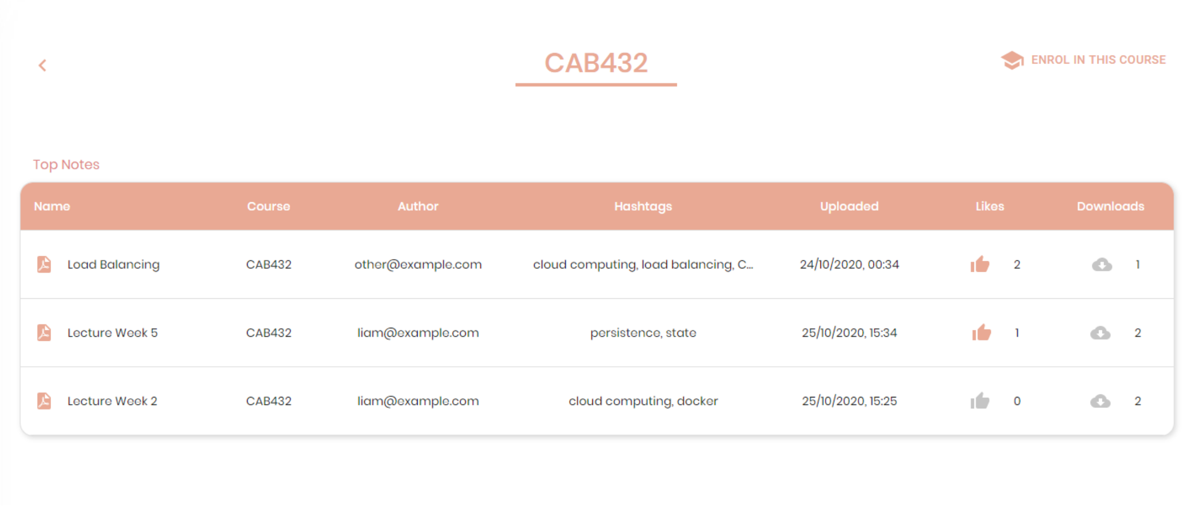
Name	Course	Author	Hashtags	Uploaded	Likes	Downloads
My New Note	CAB320	newuser@example.com	artificial intelligence, mathematics	25/10/2020, 22:23	1	0

View Courses

By clicking the courses button in the navigation bar, you can access a list of all the available courses within the system.



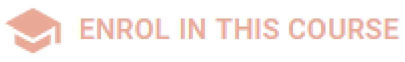
By clicking on a course, you will see a table of all the notes available in this course, ordered by likes. You can download a note by clicking on its row in the table

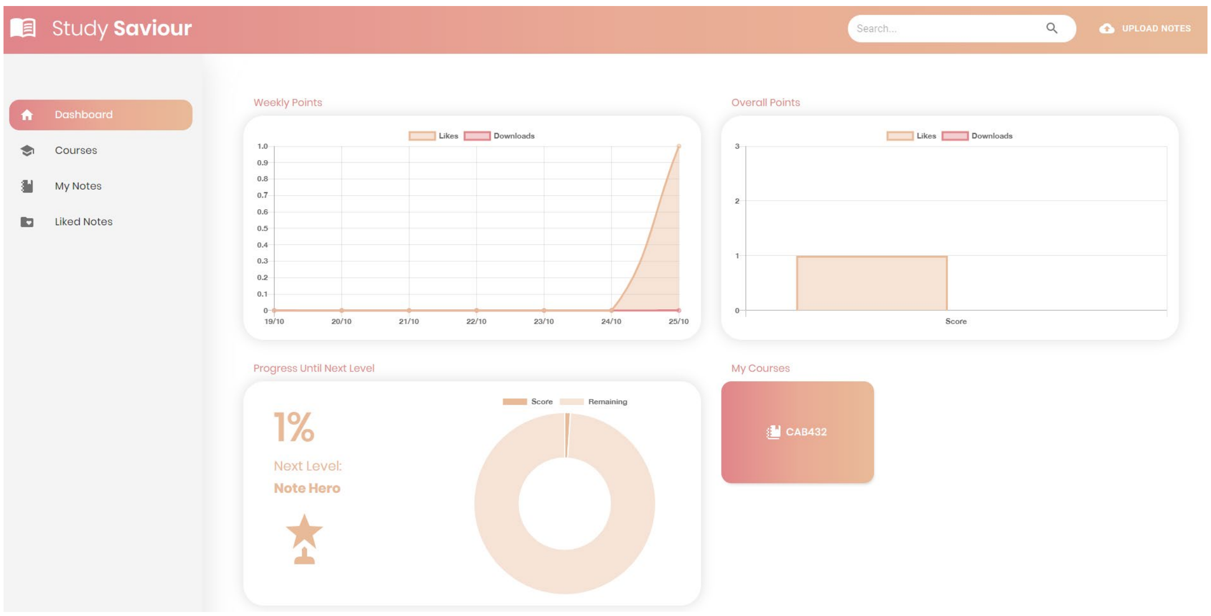


Name	Course	Author	Hashtags	Uploaded	Likes	Downloads
Load Balancing	CAB432	other@example.com	cloud computing, load balancing, C...	24/10/2020, 00:34	2	1
Lecture Week 5	CAB432	liam@example.com	persistence, state	25/10/2020, 15:34	1	2
Lecture Week 2	CAB432	liam@example.com	cloud computing, docker	25/10/2020, 15:25	0	2

Enrol in A Course

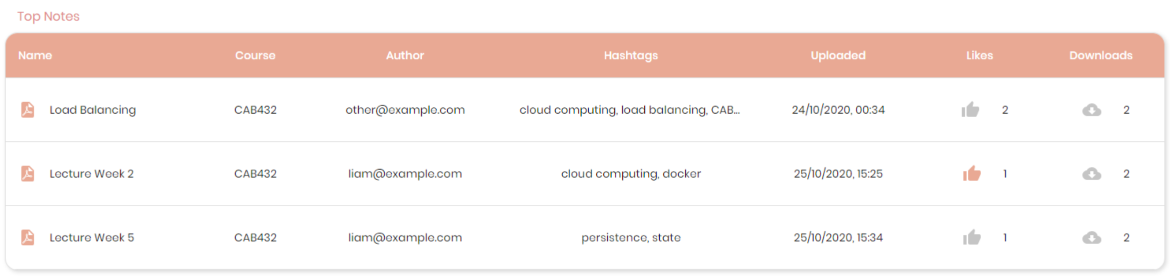
If you frequently access a particular course, you can enrol in it using the enrol in this course button. This will add the course to your list of enrolled courses, visible on your dashboard.





Like Notes

If you like a note, you can let the author know by clicking the like button next to the note in the note table.

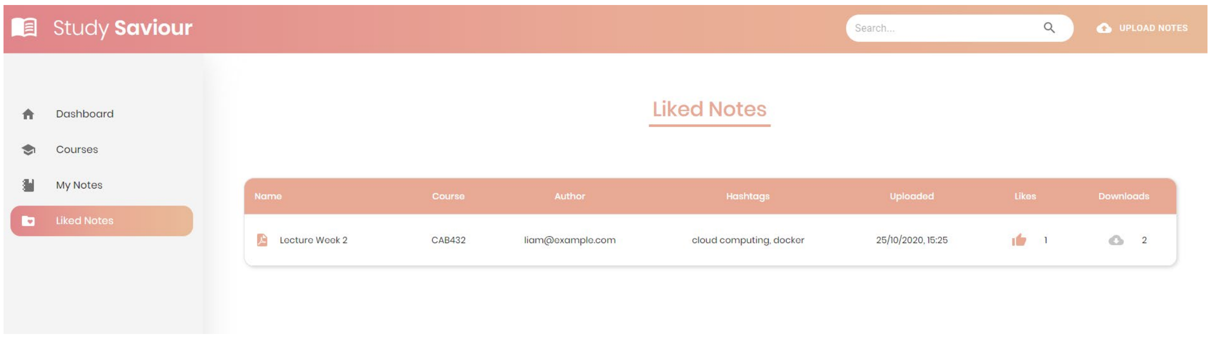


Top Notes

Name	Course	Author	Hashtags	Uploaded	Likes	Downloads
Load Balancing	CAB432	other@example.com	cloud computing, load balancing, CAB...	24/10/2020, 00:34	2	2
Lecture Week 2	CAB432	liam@example.com	cloud computing, docker	25/10/2020, 15:25	1	2
Lecture Week 5	CAB432	liam@example.com	persistence, state	25/10/2020, 15:34	1	2

View Liked Notes

You can view a list of notes that you have liked by accessing the liked notes page

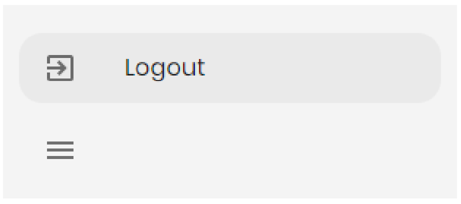


Liked Notes

Name	Course	Author	Hashtags	Uploaded	Likes	Downloads
Lecture Week 2	CAB432	liam@example.com	cloud computing, docker	25/10/2020, 15:25	1	2

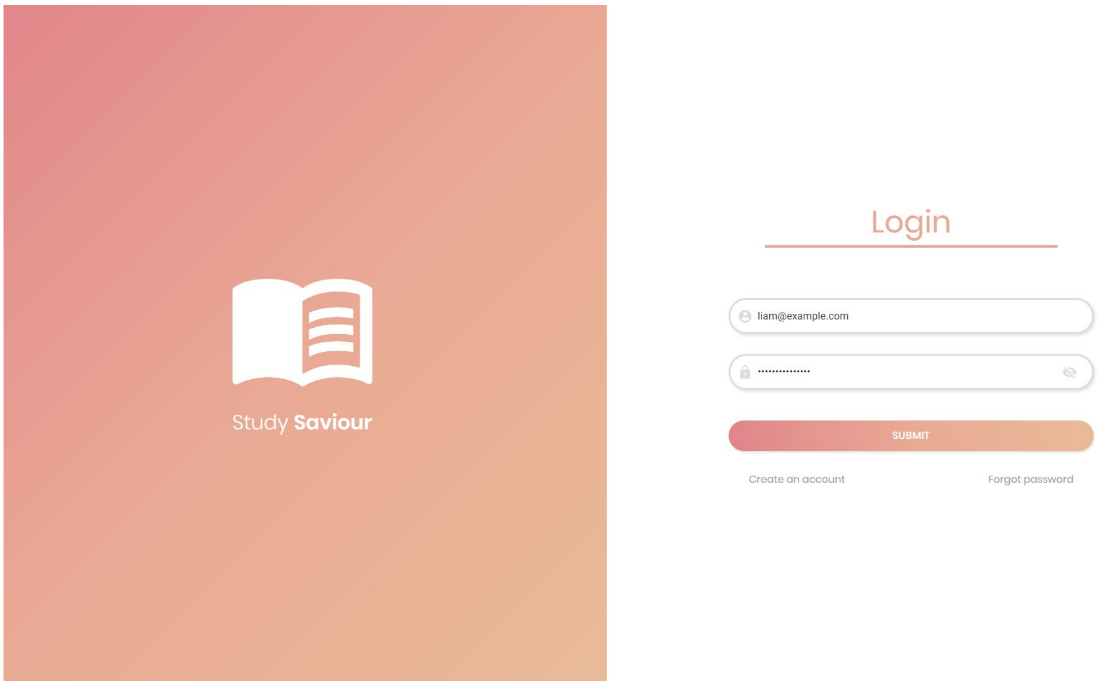
Logout

Once you have finished with the application, you can logout by clicking the logout button in the navigation bar

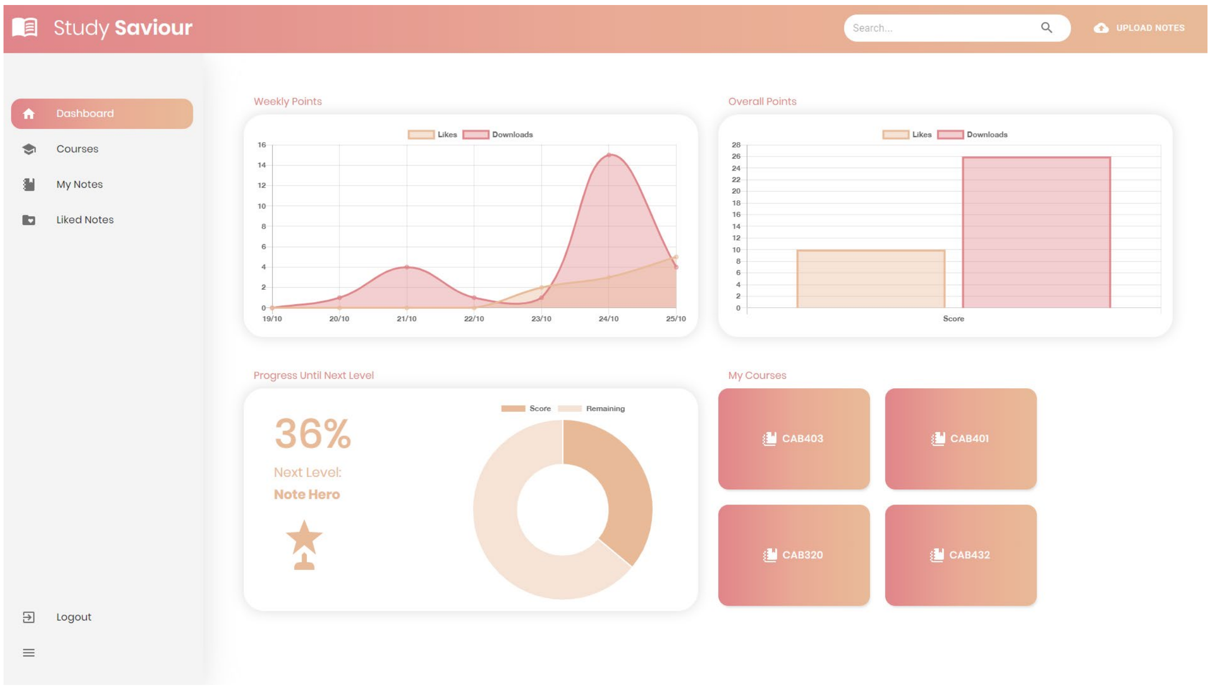


Appendices

1 Login

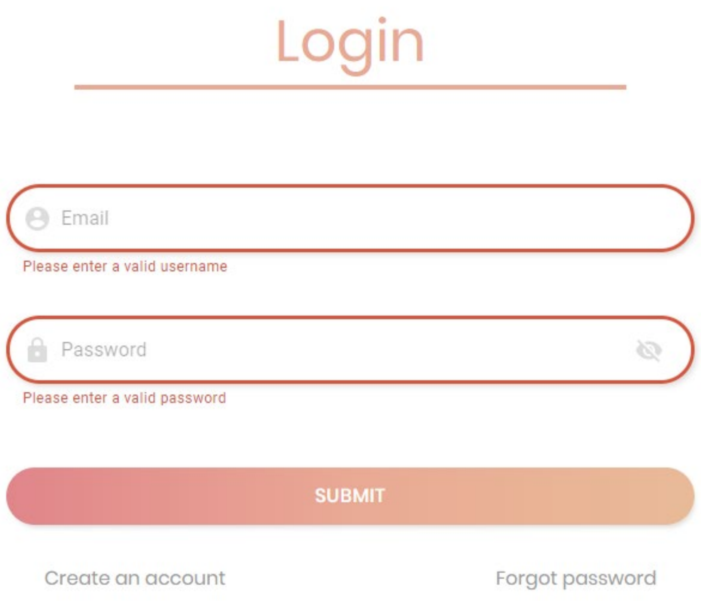


Login screen



User has successfully logged in

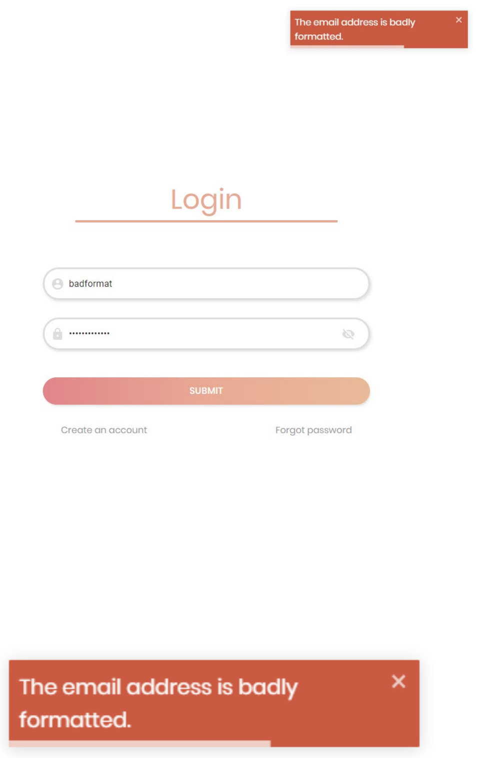
2 Handle Login Error



Please enter a valid username

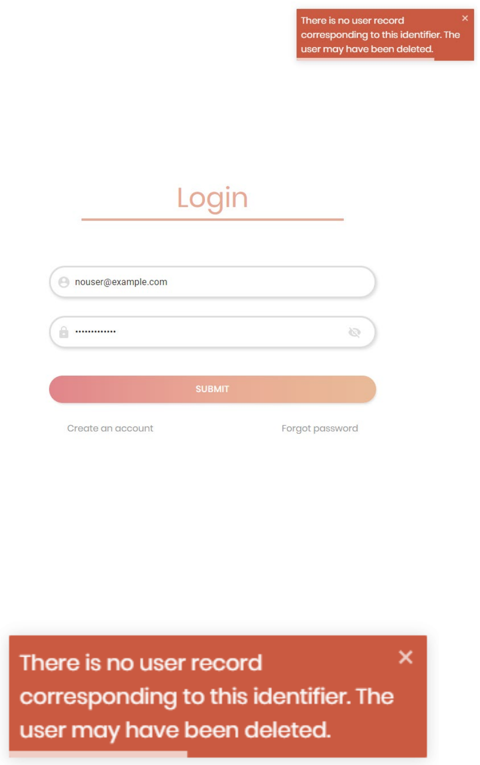
Please enter a valid password

Invalid Input: No text entered



The email address is badly formatted.

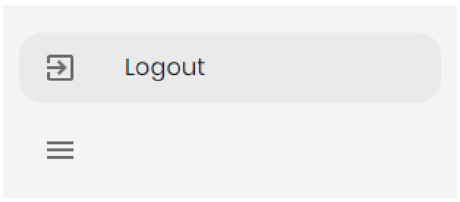
Invalid Input: Badly formatted email

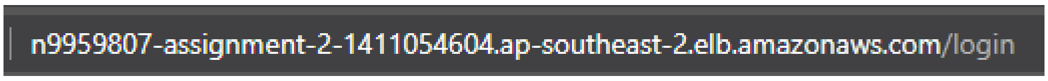


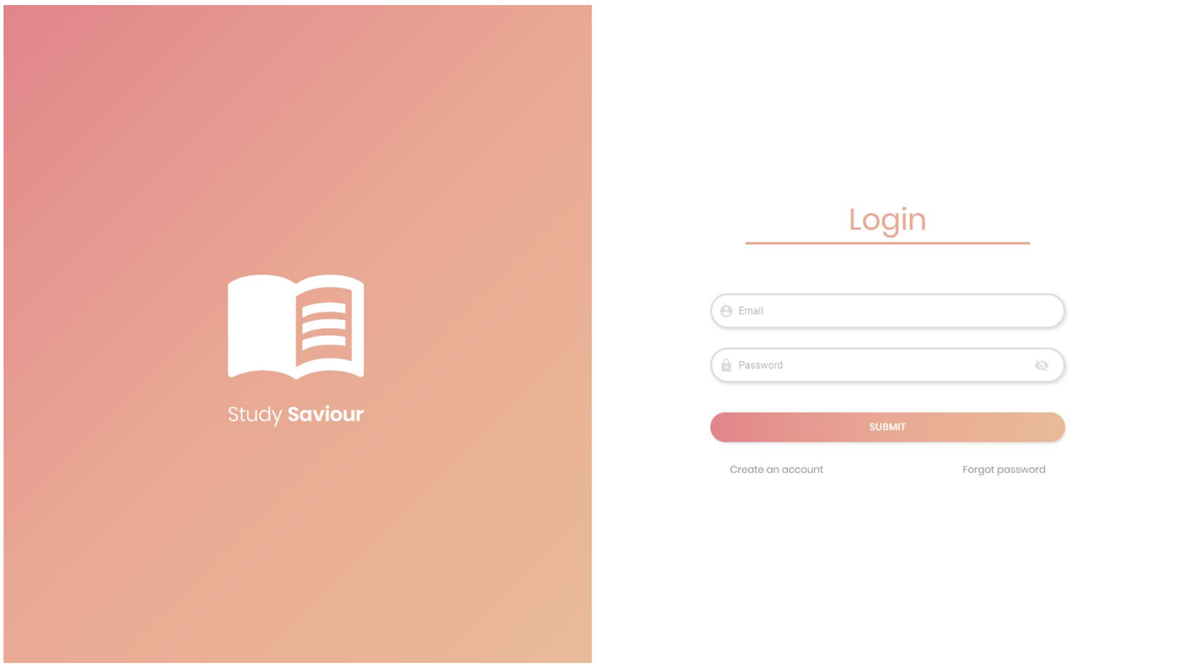
There is no user record corresponding to this identifier. The user may have been deleted.

Invalid Input: No User Record

3 Logout

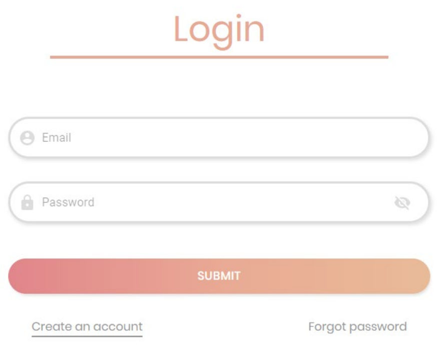


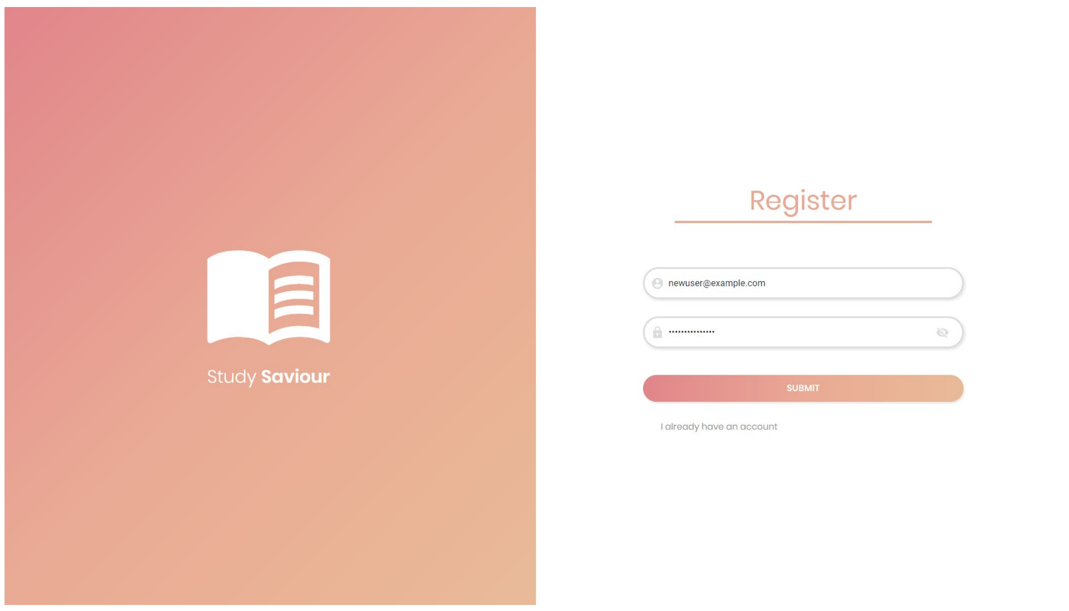

n9959807-assignment-2-1411054604.ap-southeast-2.elb.amazonaws.com/login



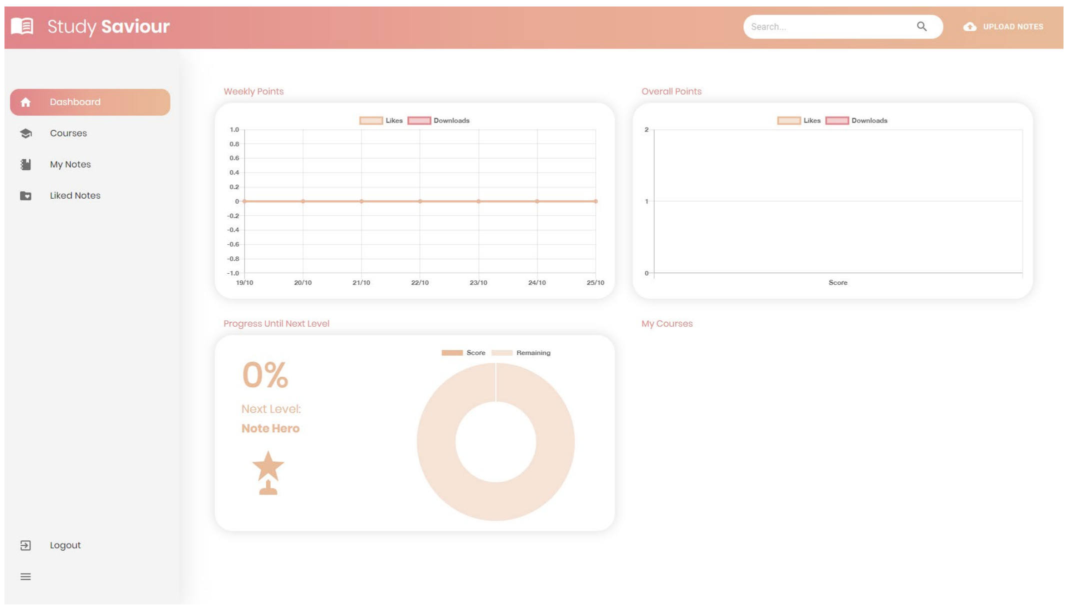
Logout button takes users back to the login screen.

4 Create Account



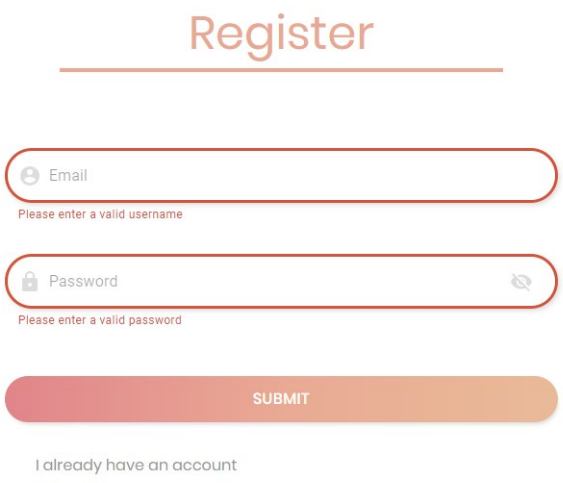


newuser@example.com



Create account button takes users to a register page. Upon successful registration, the user is logged in and directed to their dashboard

5 Handle Create Account Errors



Register

Email

Please enter a valid username

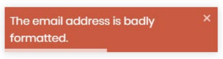
Password

Please enter a valid password

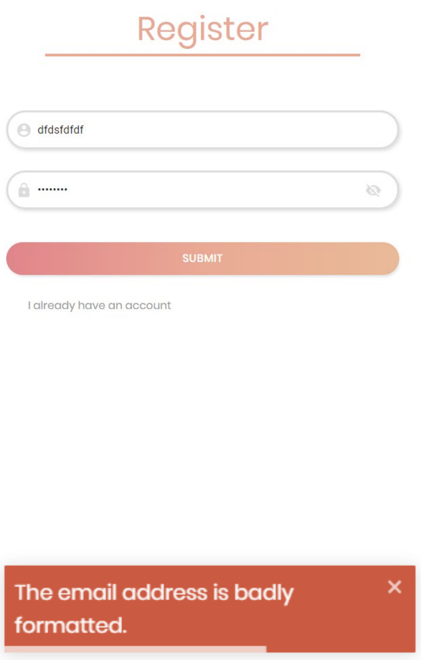
SUBMIT

I already have an account

Invalid Input: No text entered



The email address is badly formatted.



Register

dfdsfdfdf

••••••••

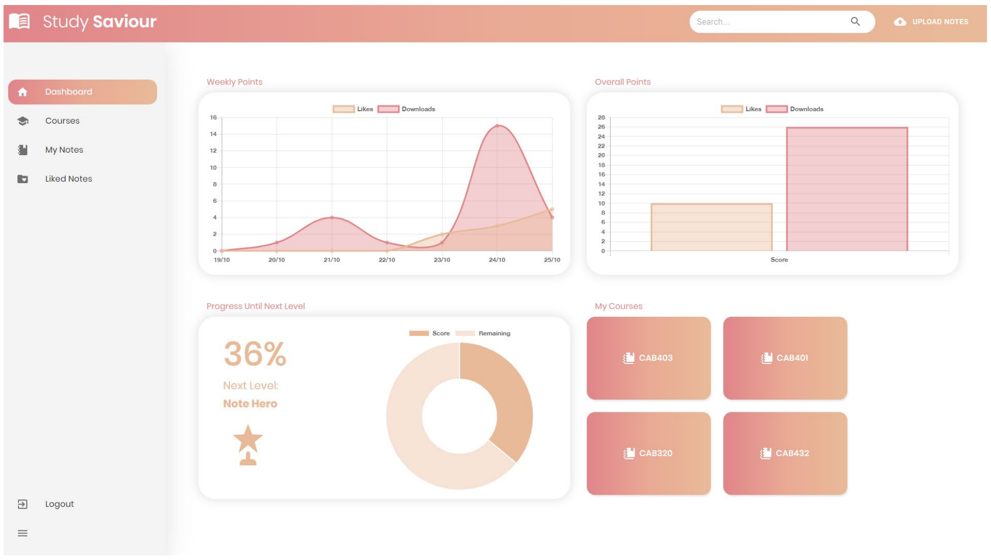
SUBMIT

I already have an account

The email address is badly formatted.

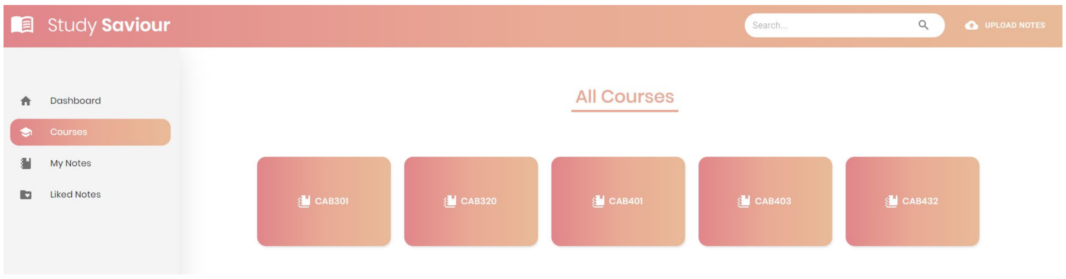
Invalid Input: Badly formatted email

6 Load Dashboard



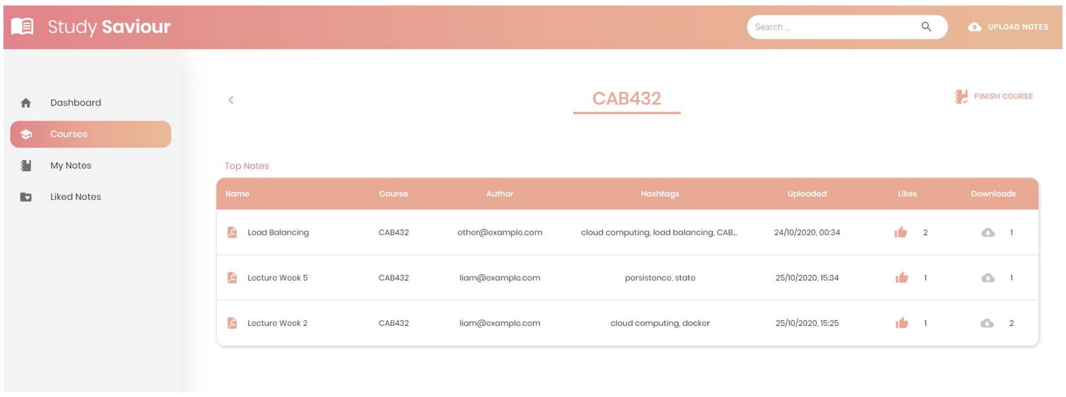
Dashboard loaded

7 Load Courses



Courses Loaded

8 Load Course

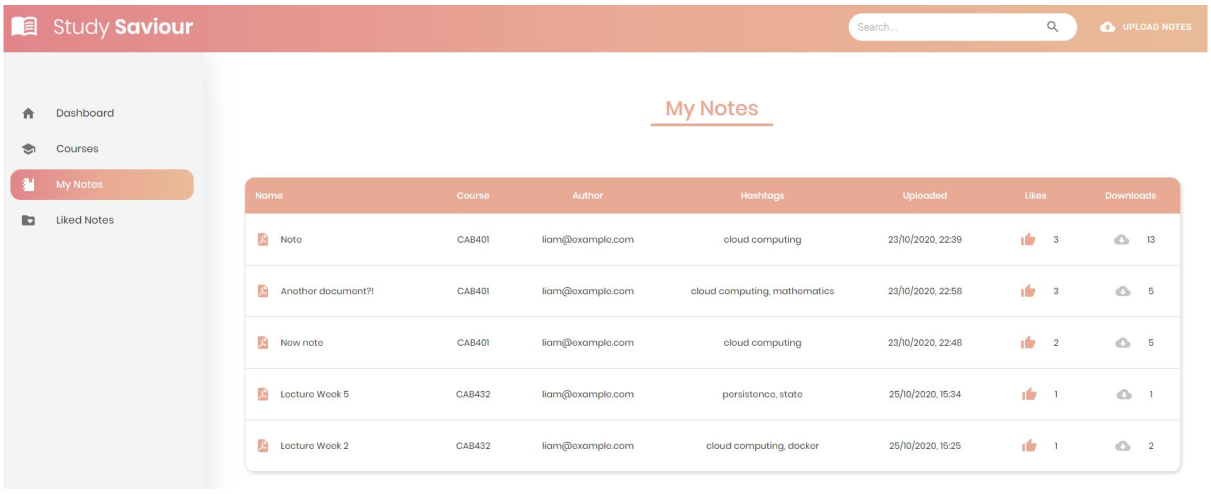


Top Notes

Name	Course	Author	Hashtags	Uploaded	Likes	Downloads
Load Balancing	CAB432	other@example.com	cloud computing, load balancing, CAB...	24/10/2020, 00:34	2	1
Lecture Week 5	CAB432	liam@example.com	persistence, state	25/10/2020, 15:34	1	1
Lecture Week 2	CAB432	liam@example.com	cloud computing, docker	25/10/2020, 15:25	1	2

Course Loaded

9 Load My Notes

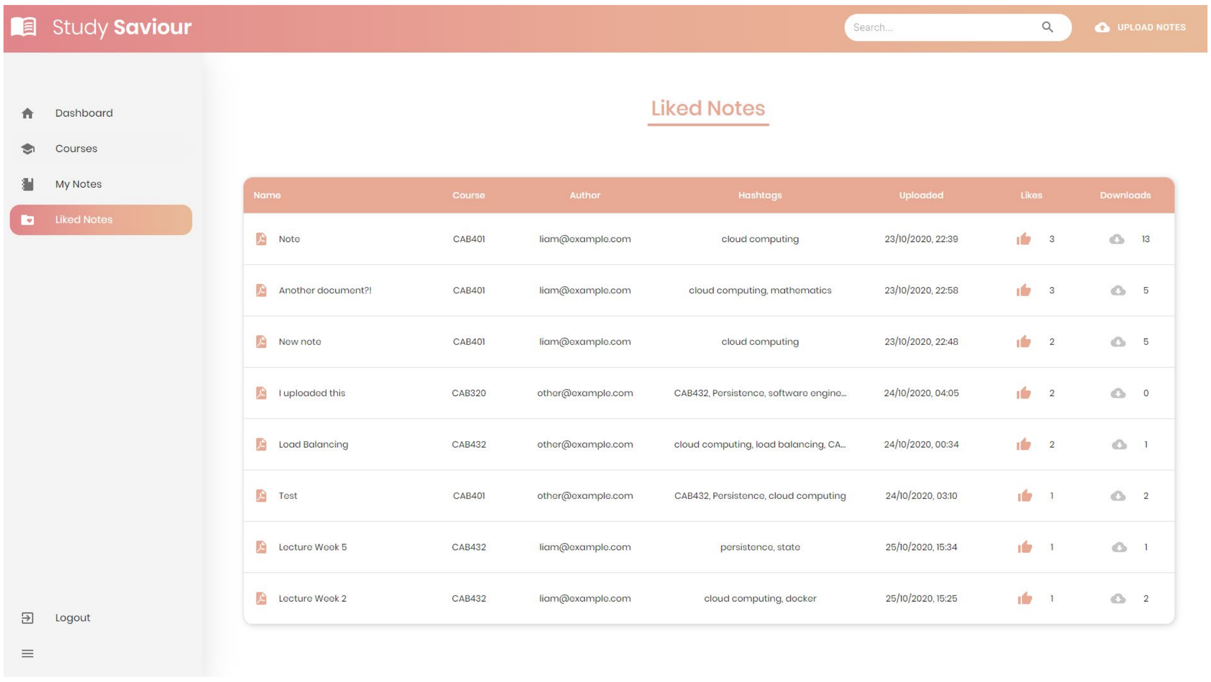


My Notes

Name	Course	Author	Hashtags	Uploaded	Likes	Downloads
Note	CAB401	liam@example.com	cloud computing	23/10/2020, 22:39	3	13
Another document?!	CAB401	liam@example.com	cloud computing, mathematics	23/10/2020, 22:58	3	5
New note	CAB401	liam@example.com	cloud computing	23/10/2020, 22:48	2	5
Lecture Week 5	CAB432	liam@example.com	persistence, state	25/10/2020, 15:34	1	1
Lecture Week 2	CAB432	liam@example.com	cloud computing, docker	25/10/2020, 15:25	1	2

My notes loaded

10 Load Liked Notes



Liked Notes

Name	Course	Author	Hashtags	Uploaded	Likes	Downloads
Note	CAB401	liam@example.com	cloud computing	23/10/2020, 22:39	3	13
Another document?!	CAB401	liam@example.com	cloud computing, mathematics	23/10/2020, 22:58	3	5
New note	CAB401	liam@example.com	cloud computing	23/10/2020, 22:48	2	5
I uploaded this	CAB320	other@example.com	CAB432, Persistence, software engine...	24/10/2020, 04:05	2	0
Load Balancing	CAB432	other@example.com	cloud computing, load balancing, CA...	24/10/2020, 00:34	2	1
Test	CAB401	other@example.com	CAB432, Persistence, cloud computing	24/10/2020, 03:10	1	2
Lecture Week 5	CAB432	liam@example.com	persistence, state	25/10/2020, 15:34	1	1
Lecture Week 2	CAB432	liam@example.com	cloud computing, docker	25/10/2020, 15:25	1	2

Course Loaded

11 Load All Cached

Load	Response (Firestore)	Response (Cache)

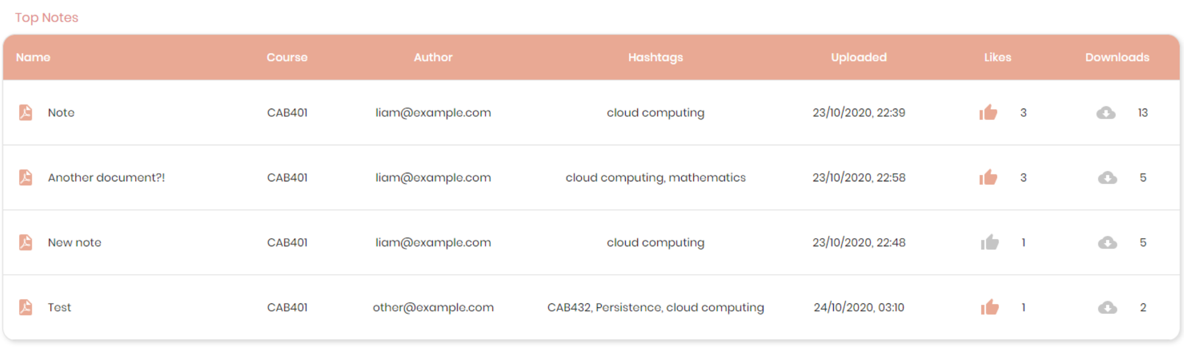
User	`User from Firestore`
	`x-response-time: 154.871ms`
		`User from Redis`
		`x-response-time: 2.547ms`

Dashboard	`Dashboard from Firetore`
	`x-response-time: 747.687ms`
		`Dashboard from Redis`
		`x-response-time: 2.108ms`

Liked Notes	`Liked notes from Firestore`
	`x-response-time: 1996.073ms`
		`Liked notes from Redis`
		`x-response-time: 2.031ms`

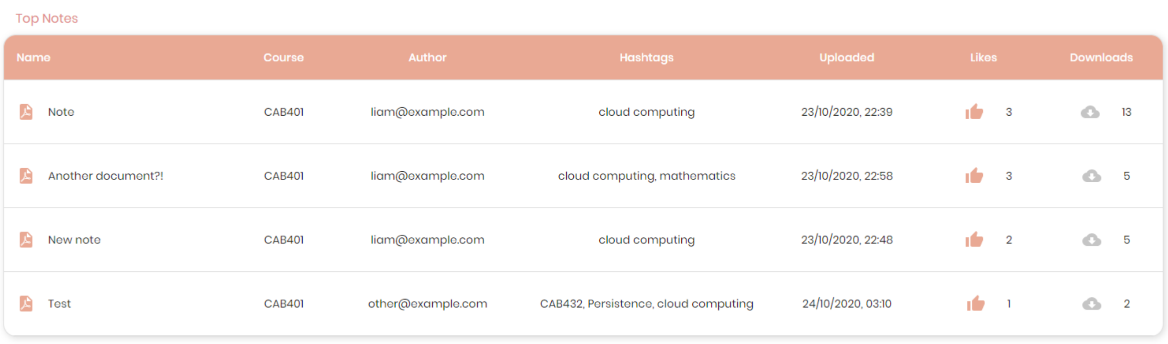
Data stored in the cache, loaded from Redis after initial load

12 Like Note



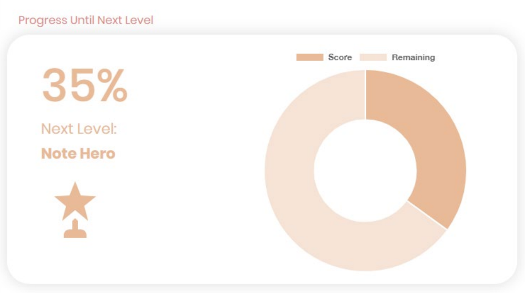
Top Notes

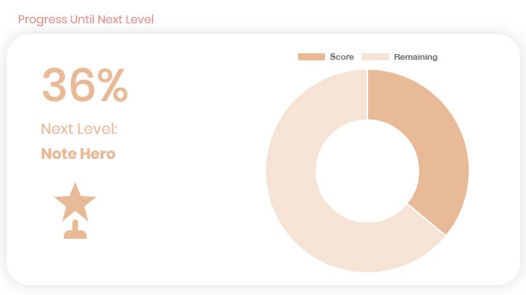
Name	Course	Author	Hashtags	Uploaded	Likes	Downloads
Note	CAB401	liam@example.com	cloud computing	23/10/2020, 22:39	3	13
Another document?!	CAB401	liam@example.com	cloud computing, mathematics	23/10/2020, 22:58	3	5
New note	CAB401	liam@example.com	cloud computing	23/10/2020, 22:48	1	5
Test	CAB401	other@example.com	CAB432, Persistence, cloud computing	24/10/2020, 03:10	1	2



Top Notes

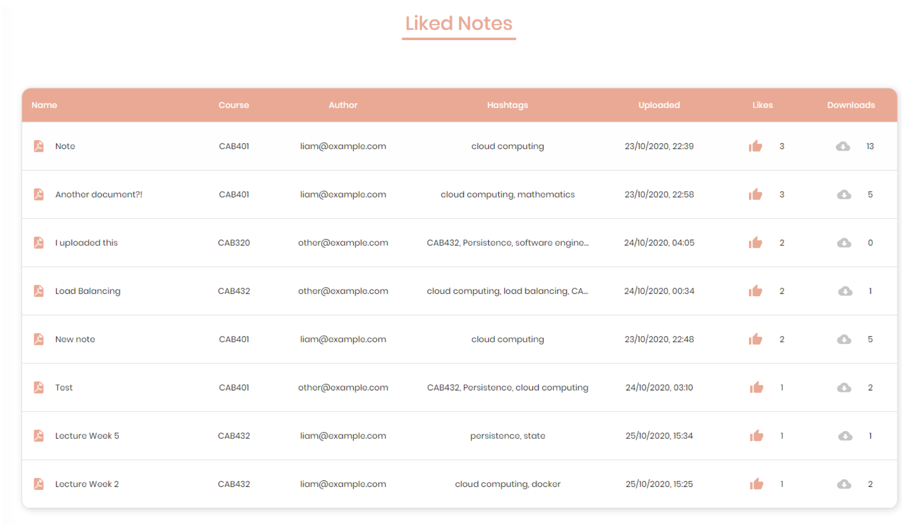
Name	Course	Author	Hashtags	Uploaded	Likes	Downloads
Note	CAB401	liam@example.com	cloud computing	23/10/2020, 22:39	3	13
Another document?!	CAB401	liam@example.com	cloud computing, mathematics	23/10/2020, 22:58	3	5
New note	CAB401	liam@example.com	cloud computing	23/10/2020, 22:48	2	5
Test	CAB401	other@example.com	CAB432, Persistence, cloud computing	24/10/2020, 03:10	1	2



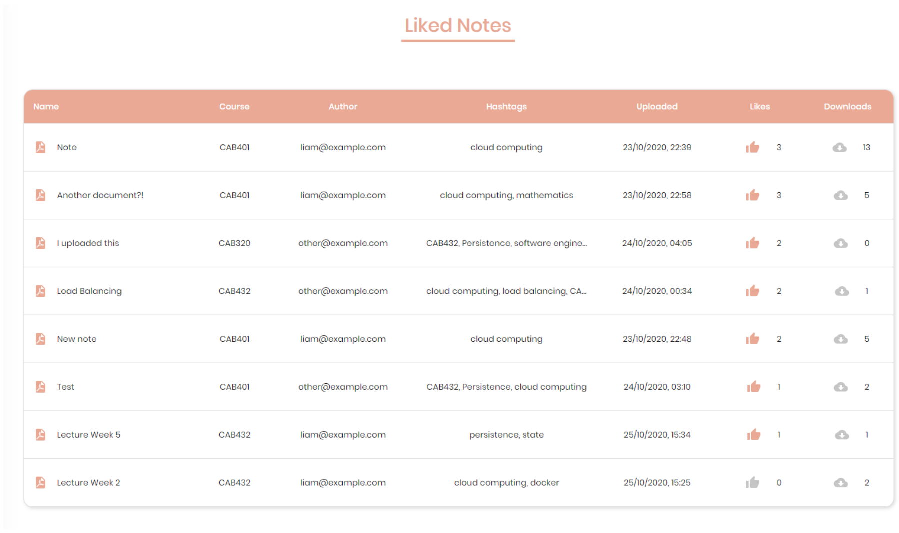


Note liked, counter increased, stats reflected

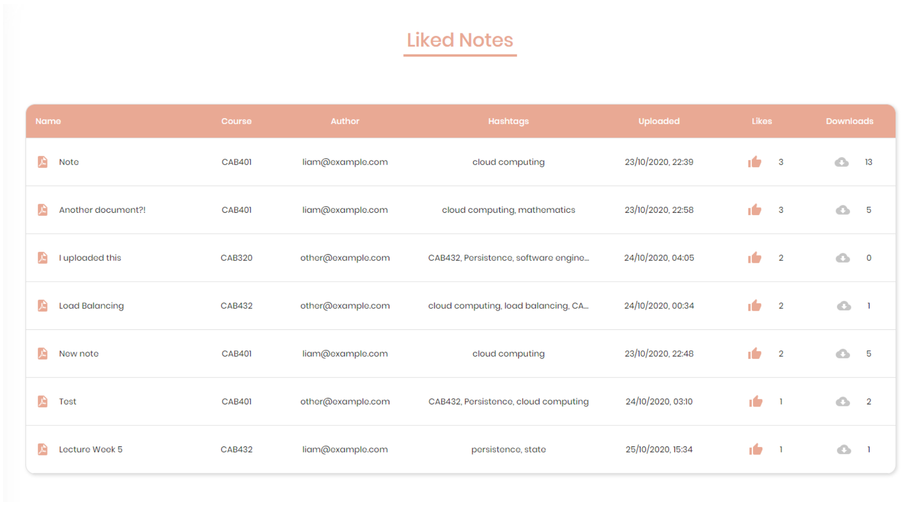
13 Unlike Note


Liked Notes

Name	Course	Author	Hashtags	Uploaded	Likes	Downloads
Note	CAB401	liam@example.com	cloud computing	23/10/2020, 22:39	3	13
Another document?!	CAB401	liam@example.com	cloud computing, mathematics	23/10/2020, 22:58	3	5
I uploaded this	CAB320	other@example.com	CAB432, Persistence, software engine...	24/10/2020, 04:05	2	0
Load Balancing	CAB432	other@example.com	cloud computing, load balancing, CA...	24/10/2020, 00:34	2	1
New note	CAB401	liam@example.com	cloud computing	23/10/2020, 22:48	2	5
Test	CAB401	other@example.com	CAB432, Persistence, cloud computing	24/10/2020, 03:10	1	2
Lecture Week 5	CAB432	liam@example.com	persistence, state	25/10/2020, 15:34	1	1
Lecture Week 2	CAB432	liam@example.com	cloud computing, docker	25/10/2020, 15:25	1	2


Liked Notes

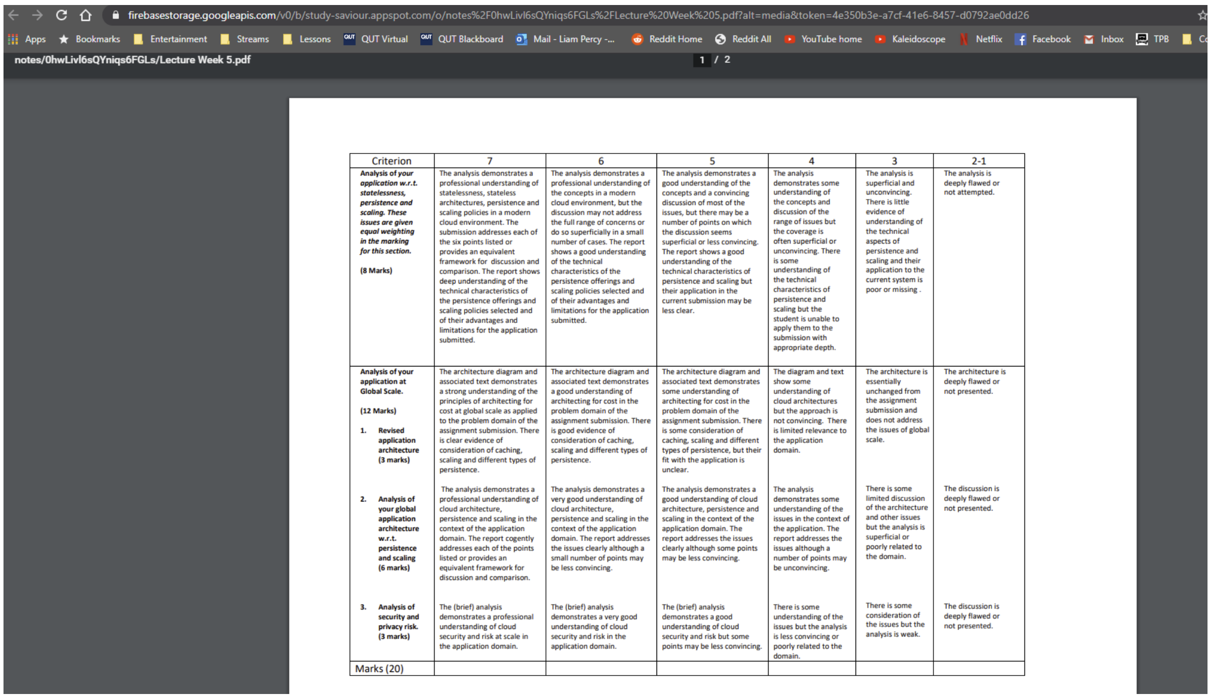
Name	Course	Author	Hashtags	Uploaded	Likes	Downloads
Note	CAB401	liam@example.com	cloud computing	23/10/2020, 22:39	3	13
Another document?!	CAB401	liam@example.com	cloud computing, mathematics	23/10/2020, 22:58	3	5
I uploaded this	CAB320	other@example.com	CAB432, Persistence, software engine...	24/10/2020, 04:05	2	0
Load Balancing	CAB432	other@example.com	cloud computing, load balancing, CA...	24/10/2020, 00:34	2	1
New note	CAB401	liam@example.com	cloud computing	23/10/2020, 22:48	2	5
Test	CAB401	other@example.com	CAB432, Persistence, cloud computing	24/10/2020, 03:10	1	2
Lecture Week 5	CAB432	liam@example.com	persistence, state	25/10/2020, 15:34	1	1
Lecture Week 2	CAB432	liam@example.com	cloud computing, docker	25/10/2020, 15:25	0	2


Liked Notes

Name	Course	Author	Hashtags	Uploaded	Likes	Downloads
Note	CAB401	liam@example.com	cloud computing	23/10/2020, 22:39	3	13
Another document?!	CAB401	liam@example.com	cloud computing, mathematics	23/10/2020, 22:58	3	5
I uploaded this	CAB320	other@example.com	CAB432, Persistence, software engine...	24/10/2020, 04:05	2	0
Load Balancing	CAB432	other@example.com	cloud computing, load balancing, CA...	24/10/2020, 00:34	2	1
New note	CAB401	liam@example.com	cloud computing	23/10/2020, 22:48	2	5
Test	CAB401	other@example.com	CAB432, Persistence, cloud computing	24/10/2020, 03:10	1	2
Lecture Week 5	CAB432	liam@example.com	persistence, state	25/10/2020, 15:34	1	1

Note unliked, counter decreases, not removed from liked notes section

14 Download Note



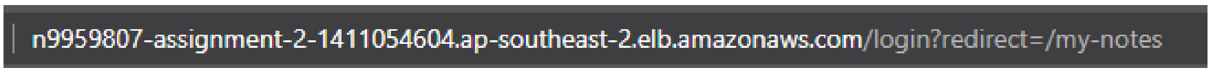
firebasestorage.googleapis.com/v0/b/study-saviour.appspot.com/o/notes%2F0hwLivl6sQYniqs6FGLs%2FLecture%20Week%205.pdf?alt=media&token=4e350b3e-a7cf-41e6-8457-d0792ae0dd26

notes/0hwLivl6sQYniqs6FGLs/Lecture Week 5.pdf 1 / 2

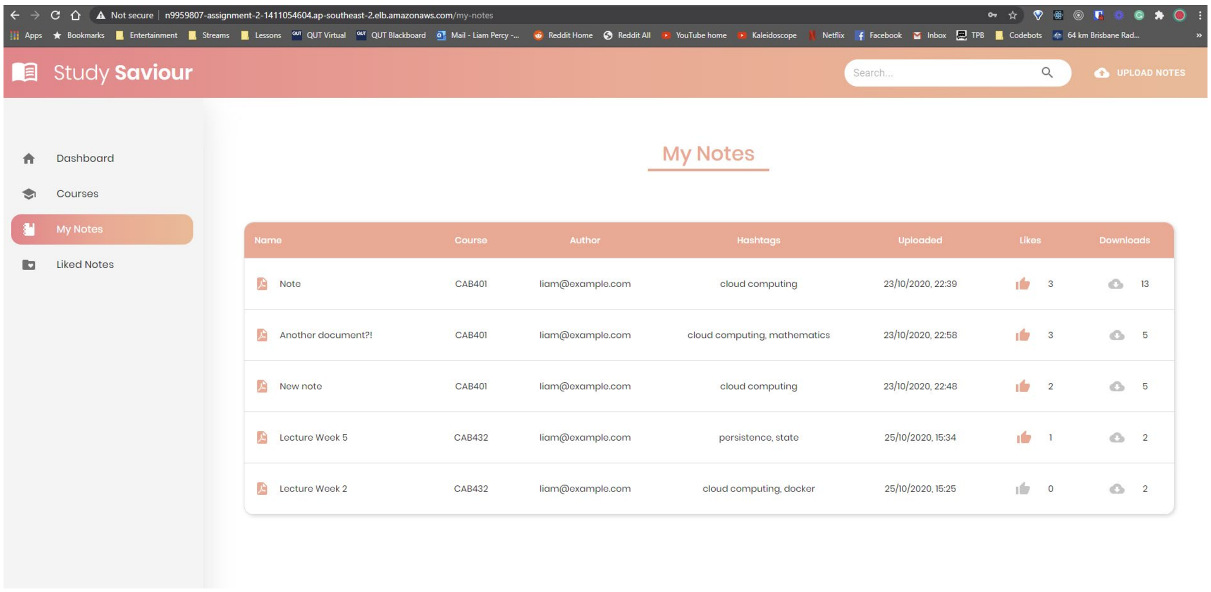
Criterion	7	6	5	4	3	2-1
Analysis of *your application w.r.t. statelessness, persistence and scaling. These issues are given equal weighting in the marking for this section.* **(8 Marks)**	The analysis demonstrates a professional understanding of statelessness, stateless architectures, persistence and scaling policies in a modern cloud environment. The submission addresses each of the six points listed or provides an equivalent framework for discussion and comparison. The report shows deep understanding of the technical characteristics of the persistence offerings and scaling policies selected and of their advantages and limitations for the application submitted.	The analysis demonstrates a professional understanding of the concepts in a modern cloud environment, but the discussion may not address the full range of concerns or do so superficially in a small number of cases. The report shows a good understanding of the technical characteristics of the persistence offerings and scaling policies selected and of their advantages and limitations for the application submitted.	The analysis demonstrates a good understanding of the concepts and a convincing discussion of most of the issues, but there may be a number of points on which the discussion seems superficial or less convincing. The report shows a good understanding of the technical characteristics of persistence and scaling but their application in the current submission may be less clear.	The analysis demonstrates some understanding of the concepts and discussion of the range of issues but the coverage is often superficial or unconvincing. There is some understanding of the technical characteristics of persistence and scaling but the student is unable to apply them to the submission with appropriate depth.	The analysis is superficial and unconvincing. There is little evidence of understanding of the technical aspects of persistence and scaling and their application to the current system is poor or missing .	The analysis is deeply flawed or not attempted.
Analysis of your application at Global Scale. **(12 Marks)** **1. Revised application architecture (3 marks)**	The architecture diagram and associated text demonstrates a strong understanding of the principles of architecting for cost at global scale as applied to the problem domain of the assignment submission. There is clear evidence of consideration of caching, scaling and different types of persistence.	The architecture diagram and associated text demonstrates a good understanding of architecting for cost in the problem domain of the assignment submission. There is good evidence of consideration of caching, scaling and different types of persistence.	The architecture diagram and associated text demonstrates some understanding of architecting for cost in the problem domain of the assignment submission. There is some consideration of caching, scaling and different types of persistence, but their fit with the application is unclear.	The diagram and text show some understanding of cloud architectures but the approach is not convincing. There is limited relevance to the application domain.	The architecture is essentially unchanged from the assignment submission and does not address the issues of global scale.	The architecture is deeply flawed or not presented.
2. Analysis of your global application architecture w.r.t. persistence and scaling (6 marks)	The analysis demonstrates a professional understanding of cloud architecture, persistence and scaling in the context of the application domain. The report cogently addresses each of the points listed or provides an equivalent framework for discussion and comparison.	The analysis demonstrates a very good understanding of cloud architecture, persistence and scaling in the context of the application domain. The report addresses the issues clearly although a small number of points may be less convincing.	The analysis demonstrates a good understanding of cloud architecture, persistence and scaling in the context of the application domain. The report addresses the issues clearly although some points may be less convincing.	The analysis demonstrates some understanding of the issues in the context of the application. The report addresses the issues although a number of points may be unconvincing.	There is some limited discussion of the architecture and other issues but the analysis is superficial or poorly related to the domain.	The discussion is deeply flawed or not presented.
3. Analysis of security and privacy risk. (3 marks)	The (brief) analysis demonstrates a professional understanding of cloud security and risk at scale in the application domain.	The (brief) analysis demonstrates a very good understanding of cloud security and risk in the application domain.	The (brief) analysis demonstrates a good understanding of cloud security and risk but some points may be less convincing.	There is some understanding of the issues but the analysis is less convincing or poorly related to the domain.	There is some consideration of the issues but the analysis is weak.	The discussion is deeply flawed or not presented.
Marks (20)						

Note downloaded and opened in browser

15 Access Unauthorised Route



n9959807-assignment-2-1411054604.ap-southeast-2.elb.amazonaws.com/login?redirect=/my-notes



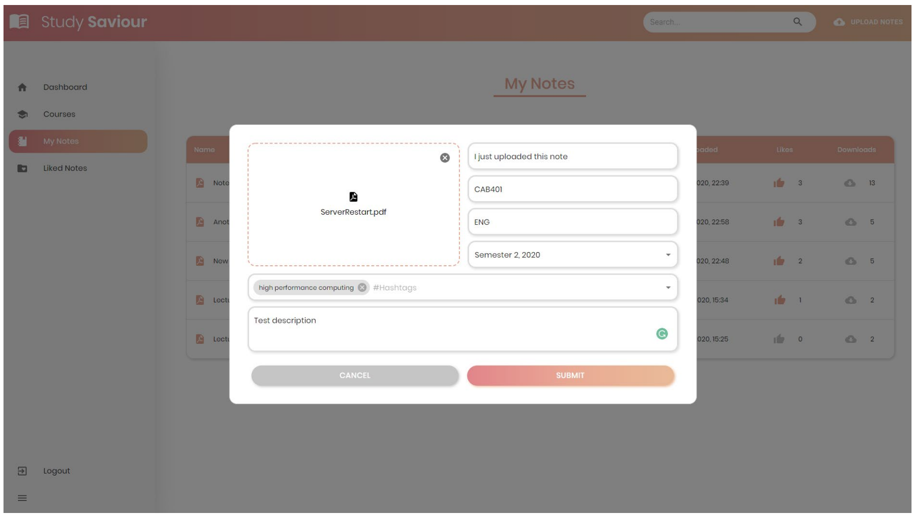
n9959807-assignment-2-1411054604.ap-southeast-2.elb.amazonaws.com/my-notes

My Notes

Name	Course	Author	Hashtags	Uploaded	Likes	Downloads
Note	CAB401	liam@example.com	cloud computing	23/10/2020, 22:39	3	13
Another document?!	CAB401	liam@example.com	cloud computing, mathematics	23/10/2020, 22:58	3	5
New note	CAB401	liam@example.com	cloud computing	23/10/2020, 22:48	2	5
Lecture Week 5	CAB432	liam@example.com	persistence, state	25/10/2020, 15:34	1	2
Lecture Week 2	CAB432	liam@example.com	cloud computing, docker	25/10/2020, 15:25	0	2

User routed to login page with query param for requested route. Upon successful login, the user is directed to the original route

16 Upload Note



ServerRestart.pdf

I just uploaded this note

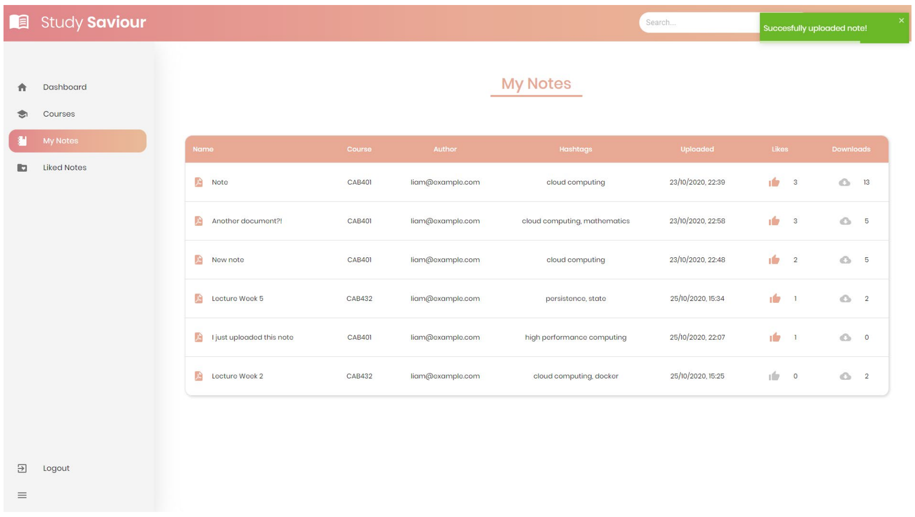
CAB401

ENG

Semester 2, 2020

high performance computing #Hashtags

Test description

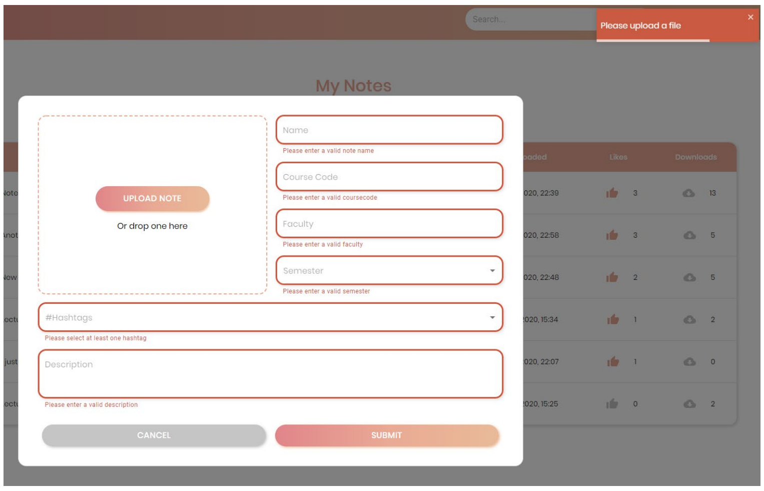


Succesfully uploaded note!

Name	Course	Author	Hashtags	Uploaded	Likes	Downloads
Note	CAB401	liam@example.com	cloud computing	23/10/2020, 22:39	3	13
Another document?!	CAB401	liam@example.com	cloud computing, mathematics	23/10/2020, 22:58	3	5
New note	CAB401	liam@example.com	cloud computing	23/10/2020, 22:48	2	5
Lecture Week 5	CAB432	liam@example.com	persistence, state	25/10/2020, 15:34	1	2
I just uploaded this note	CAB401	liam@example.com	high performance computing	25/10/2020, 22:07	1	0
Lecture Week 2	CAB432	liam@example.com	cloud computing, docker	25/10/2020, 15:25	0	2

Note uploaded

17 Handle Upload Note Errors



Please upload a file

Please enter a valid note name

Please enter a valid coursecode

Please enter a valid faculty

Please enter a valid semester

Please select at least one hashtag

Please enter a valid description

Error messages displayed