

# Software Development Practice (SWE-573)



Boğaziçi University

## Project Report

Project Name: Infrasave

Deployment URL: <https://dialecticsoftware.com>

Repository: <https://github.com/haydin505/SWE-573/releases/tag/v0.9>

Muhammet Hüseyin Nurullah AYDIN  
Advisor: [Suzan Üsküdarlı](#)

December 29, 2022

# Table of Contents

<b>HONOR CODE</b>	<b>3</b>
<b>Application Overview</b>	<b>4</b>
Product Management Overview	4
Technological Overview:	4
<b>User Stories</b>	<b>5</b>
Account User Stories (Completed)	5
Content User Stories (Completed)	5
Liked Content User Stories (Completed)	5
Tag User Stories (Completed)	6
Friend User Stories (Completed)	6
Search Functionality (Completed)	6
<b>Application Screenshots</b>	<b>7</b>
<b>Application Presentation</b>	<b>13</b>
<b>UML Diagram</b>	<b>14</b>
<b>Test Results</b>	<b>15</b>
<b>Environment Variables</b>	<b>18</b>
How to set environment variables?	18
<b>User Credentials For Test</b>	<b>18</b>
<b>3rd Party Software</b>	<b>19</b>
Frontend	19
Backend	19
<b>How to Build The Project for Development?</b>	<b>20</b>
Requirements	20
1. Build Frontend	20
2. Launch Database and Create Schema	20
3. Build Backend	20
Local Dev Environment With Docker	21
Prod Environment	21

## HONOR CODE

Related to the submission of all the project deliverables for the Swe573 2022 Fall semester project reported in this report, I Muhammet Hüseyin Nurullah AYDIN declare that:

- I am a student in the Software Engineering MS program at Bogazici University and am registered for Swe573 course during the 2022 Fall semester.
- All the material that I am submitting related to my project (including but not limited to the project repository, the final project report, and supplementary documents) have been exclusively prepared by myself.
- I have prepared this material individually without the assistance of anyone else with the exception of permitted peer assistance which I have explicitly disclosed in this report.

Muhammet Hüseyin Nurullah Aydin

# Application Overview

## Product Management Overview

This project has been developed for a content management and sharing platform suitable for a multi-user structure. Users may create content and decide who can see the created content.

### **Content visibility consist three types:**

1. Everyone: Everyone can see the content. In other words, the creator, friends and other non-friend users can see the content.
2. Friends: Only the users in the friend list can see the content. In other words, the creator and friends can see the post.
3. Private: Only the creator can see the content.

Moreover, users may add a content to the liked contents list so that they can easily find them. Users may add tags to contents. Users can search contents, users and contents with tags. Users may add other users as friends and manage friend relationships.

## Technological Overview:

The frontend and backend of the app are not tightly coupled. We can deploy them separately. This functionality fits perfectly with the deployment strategy as we use docker and cloud technologies to deploy our app. Moreover, we can scale our app horizontally, which means we can deploy a few more apps during peak hours and serve them easily. The architecture works much more efficiently during the deployment phase compared to other full-stack implementations. Since we don't need the front-end applications as much as the back-ends, we can deploy different numbers of frontend and backend servers. For example, we can deploy 2 frontend applications and 10 backend applications separately. In addition, since we developed the backend application as a REST API, we can easily integrate mobile applications or desktop applications.

### **Frontend:**

- React.JS
- Redux
- React Router Dom

### **Backend:**

- Spring Boot Starter (REST application)
- Hibernate as ORM
- MySQL

# User Stories

## Account User Stories (Completed)

1. As a user I want to login so that I can see my content.
2. As a user I want to register so that I can login and perform operations in the application.
3. As a user I want to logout so that I can securely terminate my session.
4. As a user I want to change my password so that I can keep my account secure.
5. As a user I want to change my personal details so that I can keep my profile up to date.
6. As a user I want to reset my password so that I can recover my account.

## Content User Stories (Completed)

1. As a user I want to create content.
2. As a user I want to modify content so that I can keep content up to date.
3. As a user I want to delete content.
4. As a user I want to see the contents created by me.
5. As a user, I would like to see my feed which is the contents visible to everyone, the contents only visible to friends, and the content only visible to me.
6. As a user I want to add title to my content.
7. As a user I want to add image to my content.
8. As a user I want to see when a content created and when a content modified.
9. As a user I want to add a description so that me and other people can understand the purpose of the content.
10. As a user I want to create private content so that only I can see the content.
11. As a user I want to create content only visible to my friends.
12. As a user I want to create content visible to everybody.
13. As a user I want to add tags to content so that it can be listed easily and categorized according to tag.
14. As a user I want to see the creator of the content so that I can see other posts of creator and add friends.

## Liked Content User Stories (Completed)

1. As a user I want to save a content to liked contents so that I can find it easily.
2. As a user I want to delete a content from my liked contents.

## Tag User Stories (Completed)

1. As a user I want to create a new tag so that I can organize my posts easily.
2. As a user I want to assign color to tags so that I can recognize them easily.

## Friend User Stories (Completed)

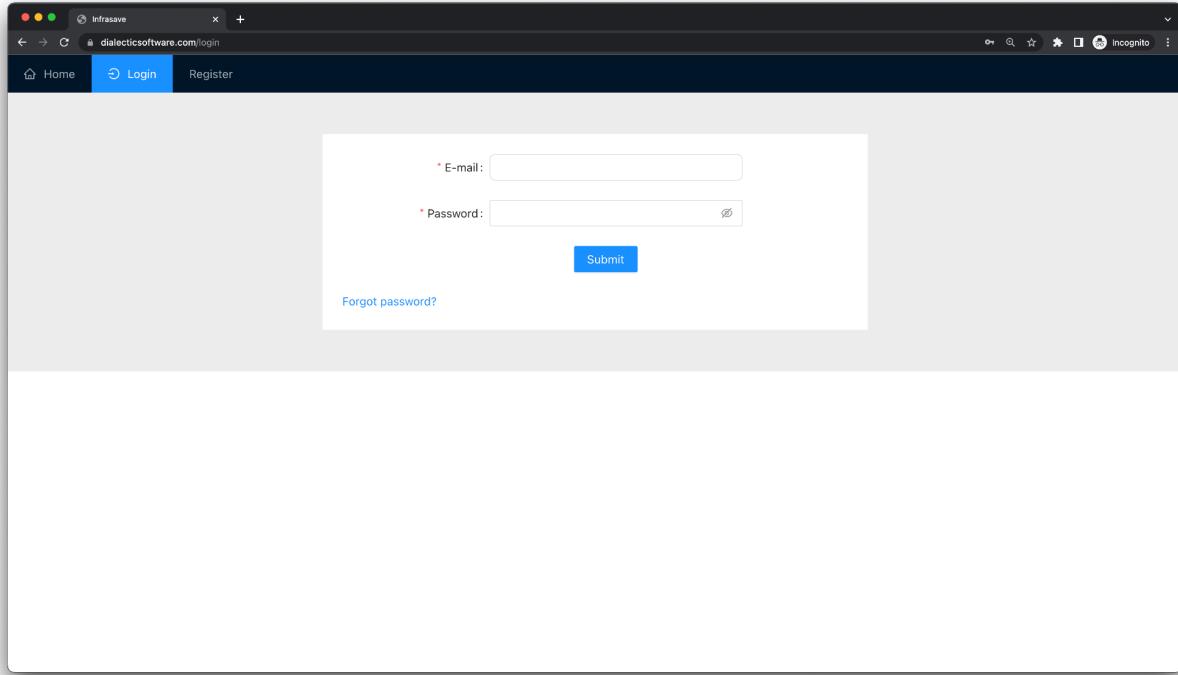
1. As a user I want to add another user as a friend so that I can see the contents only visible to friends.
2. As a user, I would like to see pending friend requests and approve or decline a friend request so that I can manage my friend list and determine who can view my contents that only my friends can view.
3. As a user, I would like to remove another user from my friend list so that I can manage my friend list.

## Search Functionality (Completed)

1. As a user I want to search for users with the combinations of name, surname, username so that I can easily find users and check out their contents.
2. As a user I want to search for contents with the combinations of title, description so that I can find the contents efficiently.
3. As a user I want to search for contents with tags attached to it so that I can find contents based on the tag.
4. As a user I want to search everything with a single button so that I can see related users, contents and tag attached contents. For example I would like to search for “politics” and check out politics related contents, contents containing “politics” tag, users whose name or surname or username of the user contains “politics”.

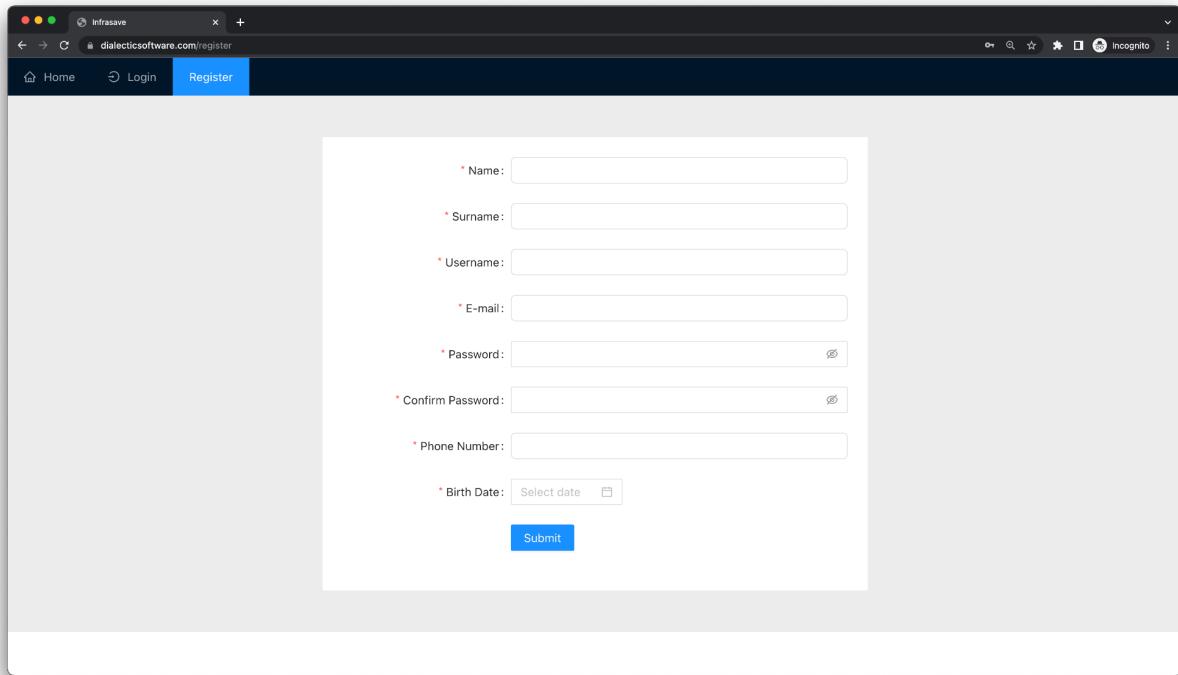
# Application Screenshots

Login Page:



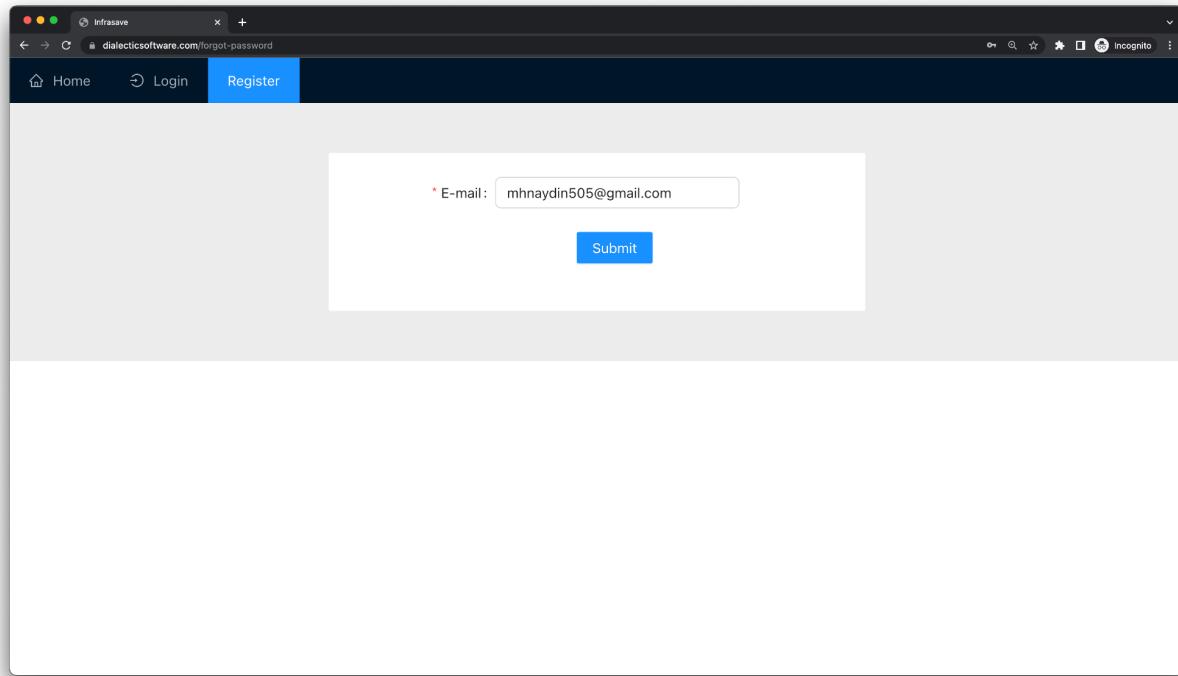
A screenshot of a web browser window showing the login page. The URL in the address bar is `dialecticsoftware.com/login`. The page has a dark header with three navigation links: "Home" (disabled), "Login" (selected and highlighted in blue), and "Register". Below the header is a form with two input fields: "E-mail:" and "Password:", both marked with a red asterisk indicating they are required. There is also a "Submit" button and a "Forgot password?" link at the bottom of the form.

Register Page:

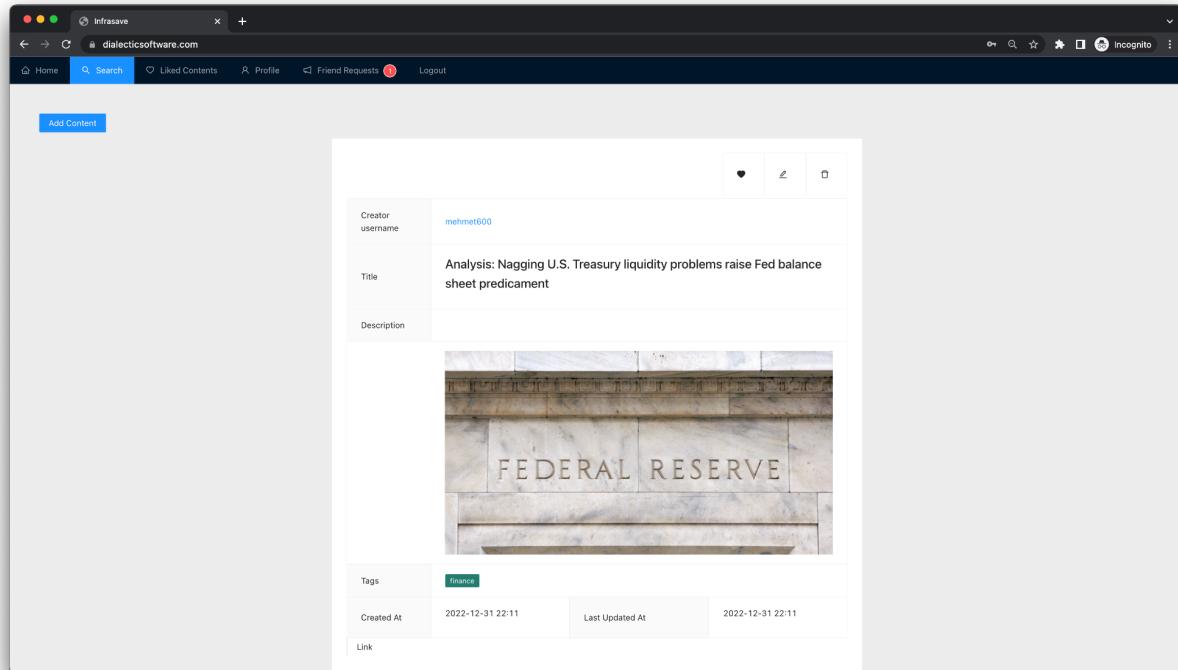


A screenshot of a web browser window showing the register page. The URL in the address bar is `dialecticsoftware.com/register`. The page has a dark header with three navigation links: "Home" (disabled), "Login" (disabled), and "Register" (selected and highlighted in blue). Below the header is a form with eight input fields, each marked with a red asterisk: "Name:", "Surname:", "Username:", "E-mail:", "Password:", "Confirm Password:", "Phone Number:", and "Birth Date:" (with a "Select date" button). There is also a "Submit" button at the bottom of the form.

## Forgot Password Page:



## Home Page:



## Liked Contents:

A screenshot of a web browser displaying a liked content item. The URL in the address bar is `dialecticsoftware.com/liked-content`. The page title is "Liked Contents". The content item has the following details:

Creator username	suleyman123
Title	Best Ski Resorts
Description	Best ski resorts of the year
Tags	<a href="#">#ski</a>
Created At	2022-12-31 22:11
Last Updated At	2023-01-01 17:03
Link	(link icon)

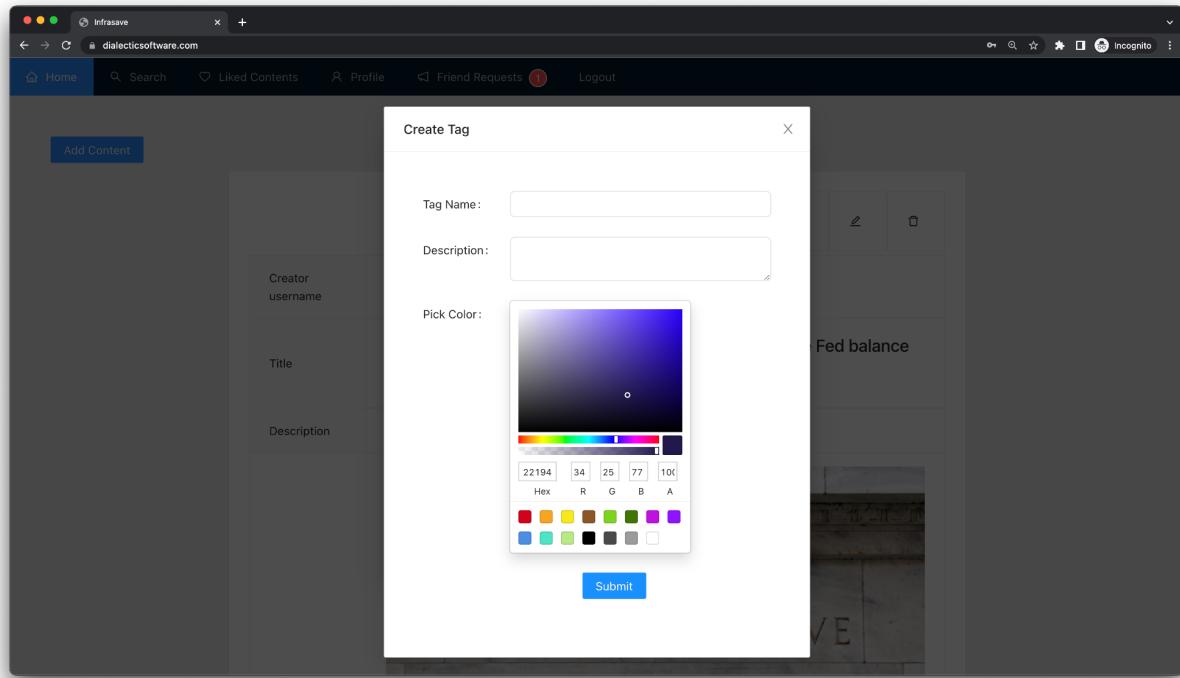
## Add Content:

A screenshot of a web browser showing an "Add Content" modal dialog. The background page has a navigation bar with links: Home, Search, Liked Contents, Profile, Friend Requests (with a notification badge), and Logout. The modal contains fields for adding a new content item:

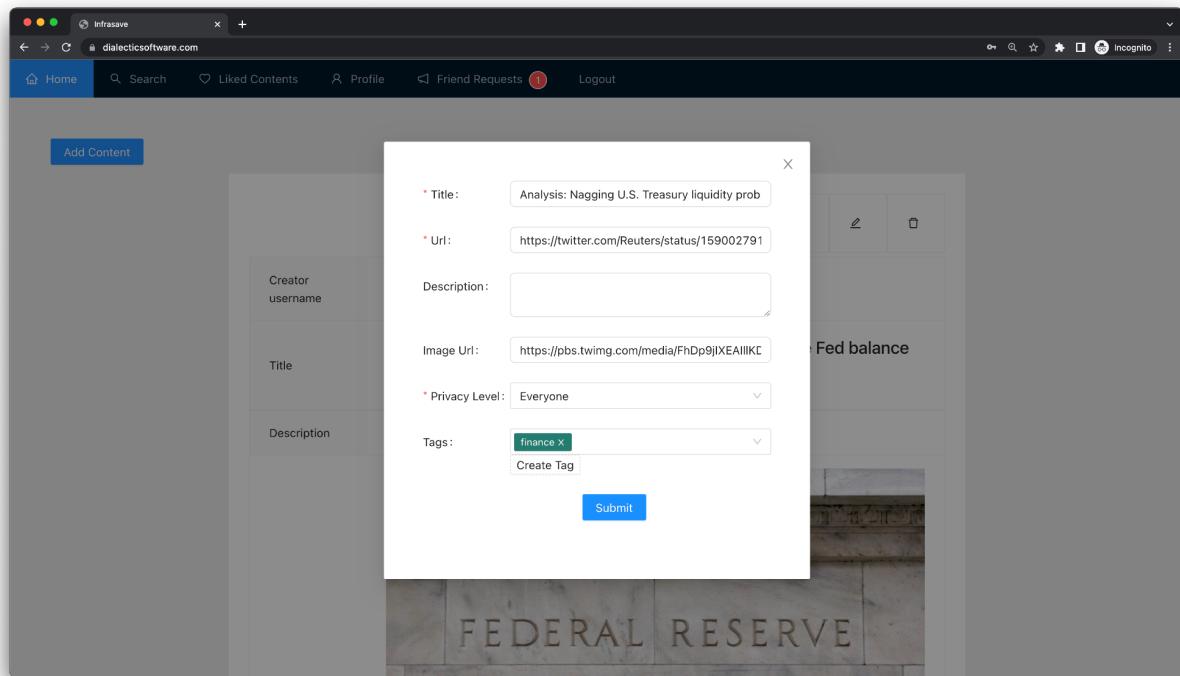
* Title:	<input type="text"/>
* Url:	<input type="text"/>
Description:	<input type="text"/>
Image Url:	<input type="text"/>
* Privacy Level:	<select></select>
Tags:	<input type="text" value="finance"/> <a href="#">Create Tag</a>

At the bottom of the modal is a "Submit" button.

## Create Tag:



## Modify Content:



## Search User:

A screenshot of a web browser showing search results for the query "meh". The search bar at the top contains "meh". Below it, there are three checkboxes: "User" (checked), "Content" (checked), and "Tag" (checked). The results table has columns: Username, Name, Surname, and Friend Status. One result is shown: "mehmet600" (Username), "Mehmet" (Name), "Selman" (Surname), and "This is your profile" (Friend Status). Navigation buttons <, 1, > are at the bottom right.

Username	Name	Surname	Friend Status
mehmet600	Mehmet	Selman	This is your profile

## Search Content:

A screenshot of a web browser showing search results for the query "tre". The search bar at the top contains "tre". Below it, there are three checkboxes: "User" (checked), "Content" (checked), and "Tag" (checked). The results table has columns: Title, Url, Created At, Last Updated At, Description, My Content, Creator User, and Tags. One result is shown: "Analysis: Nagging U.S. Treasury liquidity problems raise Fed balance sheet predicament" (Title), "Link" (Url), "2022-12-31 19:11" (Created At), "2022-12-31 19:11" (Last Updated At), "mehmet600" (Creator User), and "finance" (Tags). Navigation buttons <, 1, > are at the bottom right.

Title	Url	Created At	Last Updated At	Description	My Content	Creator User	Tags
Analysis: Nagging U.S. Treasury liquidity problems raise Fed balance sheet predicament	Link	2022-12-31 19:11	2022-12-31 19:11			mehmet600	finance

## Search Content by Tag:

The screenshot shows a web browser window with the URL [dialecticsoftware.com/search?query=finance](https://dialecticsoftware.com/search?query=finance). The page has a dark header with navigation links: Home, Search (highlighted in blue), Liked Contents, Profile, Friend Requests (with a red notification badge), and Logout. A search bar at the top contains the query "finance". Below the search bar are three checkboxes: User (checked), Content (checked), and Tag (checked). The main content area displays a table with one row of data. The table columns are: Title, Url, Created At, Last Updated At, Description, My Content, Creator User, and Tags. The single row of data is: "Analysis: Nagging U.S. Treasury liquidity problems raise Fed balance sheet predicament" (Title), "Link" (Url), "2022-12-31" (Created At), "2022-12-31 19:11" (Last Updated At), empty (Description), empty (My Content), "mehmet600" (Creator User), and "finance" (Tags). At the bottom right of the table are navigation arrows for pagination.

## Profile:

The screenshot shows a web browser window with the URL [dialecticsoftware.com/profile](https://dialecticsoftware.com/profile). The page has a dark header with navigation links: Home, Search, Liked Contents, Profile (highlighted in blue), Friend Requests (with a red notification badge), and Logout. The main content area is divided into two sections: "User Details" and "Friend List". The "User Details" section contains form fields for Name (Mehmet), Surname (Selman), Birth Date (2022-12-31), Username (mehmet600), E-mail (infrasave.app@gmail.com), and Phone Number (+905111111111). Below these fields is a "Number of Friends" field showing "0" and an "Update" button. The "Friend List" section shows a message "No data" next to a small user icon.

Profile Created Contents:

The screenshot shows a web browser window with the URL [dialecticsoftware.com/profile](https://dialecticsoftware.com/profile). The page title is "Created Contents". There is a blue button labeled "Add Content". Below it is a card containing the following information:

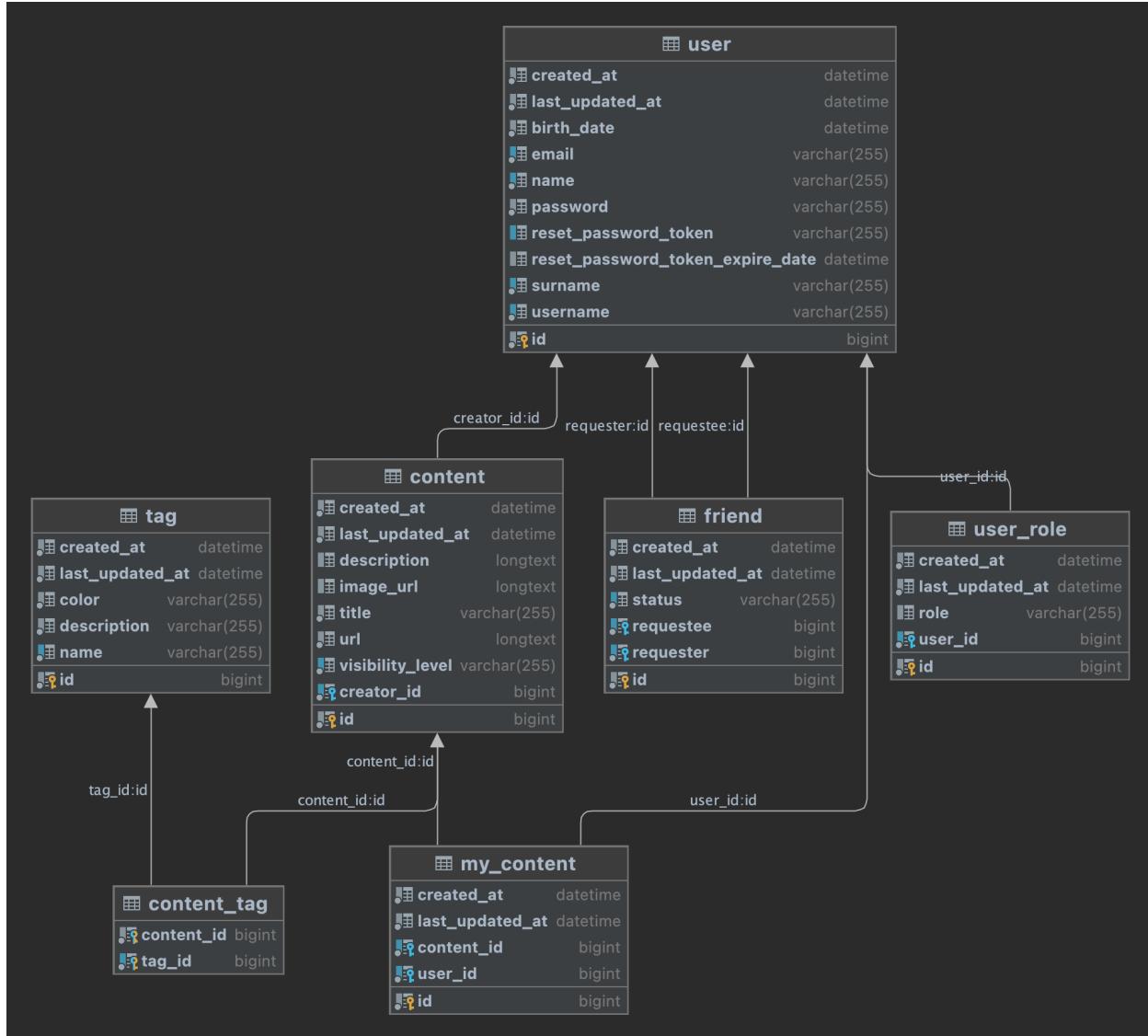
Creator username	mehmet600
Title	Analysis: Nagging U.S. Treasury liquidity problems raise Fed balance sheet predicament
Description	(empty)

Below the card is a small image of the Federal Reserve building.

## Application Presentation

Application presentation video link: <https://www.youtube.com/watch?v=i-jGzhF3Gxc>

# UML Diagram



# Test Results

I have developed integration tests for all of the functionalities in the application. These test classes will greatly help me when I modify or develop more functionality in the application.

- Improved reliability and robustness: With these Integration tests we can identify issues that may not have been discovered. By testing how different components work together, integration tests can uncover issues that may arise when the system is used in a real-world environment. That's why I have developed integration tests.
- Enhanced confidence in the system: Integration tests provide a higher level of confidence in the system's functionality and its ability to handle real-world scenarios. This can be especially important when releasing new features or making changes to the system.
- Easier debugging: Integration tests can help identify the root cause of issues more quickly, as they provide more context around how different components are interacting with each other.
- Better coverage: Integration tests can help ensure that the system is thoroughly tested and that all relevant scenarios are covered. This can help reduce the risk of defects or issues being missed during testing.

**com.infrasave in infrasave-rest-app: 25 total, 25 passed** 5.82 s

[Collapse](#) | [Expand](#)

<b>AccountIntegrationTests</b>	2.66 s
<a href="#">shouldLogin()</a>	passed 971 ms
<a href="#">shouldFailLogin()</a>	passed 201 ms
<a href="#">shouldRegister()</a>	passed 1.46 s
<a href="#">shouldFailRegister()</a>	passed 28 ms
<b>ContentIntegrationTests</b>	709 ms
<a href="#">saveUser()</a>	passed 97 ms
<a href="#">shouldGetMyFeed()</a>	passed 235 ms
<a href="#">shouldCreateContent()</a>	passed 128 ms
<a href="#">shouldUpdateContent()</a>	passed 130 ms
<a href="#">shouldDeleteContent()</a>	passed 119 ms
<b>FriendIntegrationTests</b>	1.14 s
<a href="#">shouldLogin()</a>	passed 463 ms
<a href="#">shouldSendFriendRequest()</a>	passed 122 ms
<a href="#">shouldGetPendingFriendRequest()</a>	passed 118 ms
<a href="#">shouldApprovePendingFriendRequest()</a>	passed 122 ms
<a href="#">shouldGetApprovedFriends()</a>	passed 107 ms
<a href="#">shouldRemoveFriend()</a>	passed 104 ms
<a href="#">shouldGetEmptyApprovedFriends()</a>	passed 104 ms
<b>TagIntegrationTests</b>	660 ms
<a href="#">shouldLogin()</a>	passed 286 ms
<a href="#">shouldCreateTag()</a>	passed 108 ms
<a href="#">shouldCreateContent()</a>	passed 149 ms
<a href="#">shouldGetMyFeedWithTag()</a>	passed 117 ms
<b>LikedContentIntegrationTests</b>	646 ms
<a href="#">saveUser()</a>	passed 101 ms
<a href="#">shouldCreateContent()</a>	passed 206 ms
<a href="#">shouldAddToLikedContent()</a>	passed 111 ms
<a href="#">shouldGetLikedContents()</a>	passed 121 ms
<a href="#">shouldDeleteContent()</a>	passed 107 ms

Generated by IntelliJ IDEA on 12/31/22, 9:29 PM

Current scope: all classes

## Overall Coverage Summary

Package	Class, %	Method, %	Line, %
all classes	63.8% (44/69)	34% (111/326)	25.3% (263/1039)

## Coverage Breakdown

Package ▲	Class, %	Method, %	Line, %
com.infrasave	100% (1/1)	25% (1/4)	1.3% (1/79)
com.infrasave.bean	25% (5/20)	31.2% (25/80)	28.7% (39/136)
com.infrasave.config	87.5% (7/8)	85.7% (24/28)	81.2% (69/85)
com.infrasave.controller	100% (8/8)	27% (10/37)	19.7% (28/142)
com.infrasave.controller.admin	100% (1/1)	50% (1/2)	22.2% (2/9)
com.infrasave.entity	100% (7/7)	35.6% (26/73)	37.3% (28/75)
com.infrasave.enums	66.7% (2/3)	40% (4/10)	42.9% (9/21)
com.infrasave.exception	0% (0/6)	0% (0/9)	0% (0/13)
com.infrasave.repository.content	100% (1/1)	16.7% (1/6)	5.9% (2/34)
com.infrasave.repository.friend	100% (1/1)	50% (1/2)	33.3% (2/6)
com.infrasave.repository.mycontent	100% (1/1)	33.3% (1/3)	20% (2/10)
com.infrasave.repository.tag	100% (1/1)	100% (1/1)	100% (2/2)
com.infrasave.repository.user	100% (1/1)	44.4% (4/9)	38.1% (16/42)
com.infrasave.service	80% (8/10)	19.4% (12/62)	16.4% (63/385)

generated on 2022-12-31 21:32

# Environment Variables

In order to build, test or deploy the project we must set environment variables on the operating system. That means we do not share sensitive information in the repository. Therefore, I will share environment variables below.

## How to set environment variables?

We can set environment variables by modifying bashrc file located at `~/.bashrc`. Copy and paste below code block to the file and logout and login again. Now you are all set. You can build, test, deploy the application.

```
export REACT_APP_BASE_URL=https://dialecticsoftware.com/api
export active_profile=prod
export datasource_url=
export datasource_username=
export datasource_password=
export mail_username=
export mail_password=
```

## User Credentials For Test

Infrasave Application Credentials:

User ID	E-mail	Password
1	infrasave.app@gmail.com	123456
2	test@test2.com	123456

## 3rd Party Software

In order to view the license please click the software.

Frontend

Software	License
<a href="#">React</a>	MIT License
<a href="#">Ant-Design</a>	MIT License
<a href="#">Redux</a>	MIT License
<a href="#">Axios</a>	MIT License
<a href="#">React Router Dom</a>	MIT License

Backend

Software	License
<a href="#">Java 17</a>	NFTC
<a href="#">Spring Framework</a>	Apache License 2.0
<a href="#">Hibernate</a>	LGPL 2.1
<a href="#">MySQL</a>	FOSS

# How to Build The Project for Development?

You can use make or manual commands to build the application. make command can be installed via brew install make

## Requirements

- Docker and docker-compose
- Java 17

### 1. Build Frontend

Run

```
make build-frontend
```

or

```
infrasave/infrasave-react-client/node/npm ci --prefix  
infrasave/infrasave-react-client &&  
infrasave/infrasave-react-client/node/npm start --prefix  
infrasave/infrasave-react-client
```

### 2. Launch Database and Create Schema

Run

```
make build-db
```

or

```
docker-compose -f infrasave/docker-compose-local.yml up -d db
```

### 3. Build Backend

Run

```
make build-backend
```

or

```
cd infrasave/infrasave-rest-app && ./mvnw spring-boot:run
```

Now you can access application with the following link:

- <http://localhost:3000>
- 

## Local Dev Environment With Docker

You can use this type of build process in order to test or examine the application.

- Run following command

```
docker-compose -f infrasave/docker-compose-local.yml build && \
docker-compose -f infrasave/docker-compose-local.yml create && \
docker-compose -f infrasave/docker-compose-local.yml start
```

Now you can access application with the following link: <http://localhost:3000>

## Prod Environment

- Remove local images with following command

```
docker-compose -f infrasave/docker-compose.yml down --remove-orphans --rmi
all --volumes
```

- You must set environment variables in your operations system for following properties
- Please contact owner of the repository in order to get value of the below environment variables.

```
export REACT_APP_BASE_URL=https://dialecticsoftware.com/api
export active_profile=
export datasource_url=
export datasource_username=
export datasource_password=
export mail_username=
export mail_password=
```

- Run following command

```
docker-compose -f infrasave/docker-compose.yml up -d
```

Now you can access application with the following link: <http://localhost:3000>