

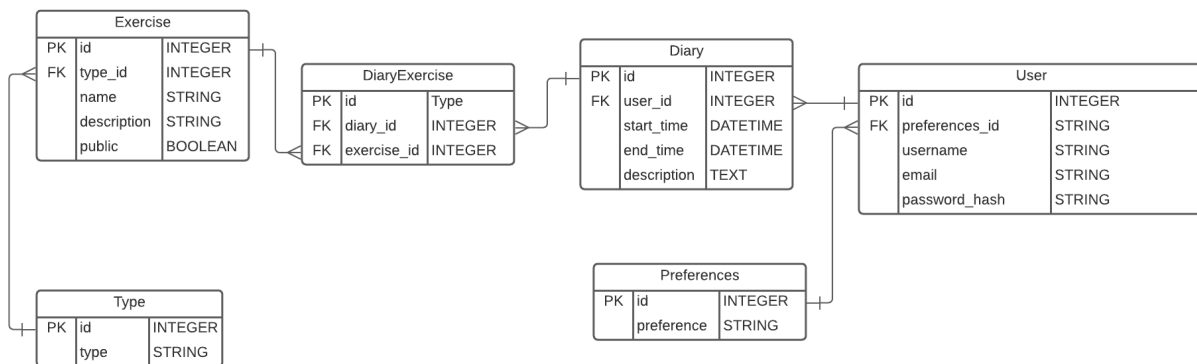
13DTP Project Planning

<https://github.com/idoYT/practicetool>

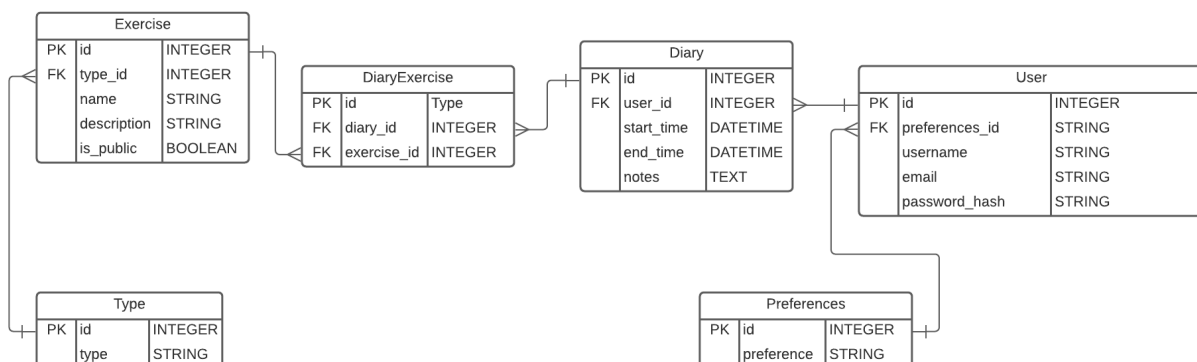
Purpose, Function, and End User of my Website.

I will be designing a diary web application, to track progress and help users reach their goals. Primarily used for tracking when the user practices and what they're doing, should help them set goals and that.

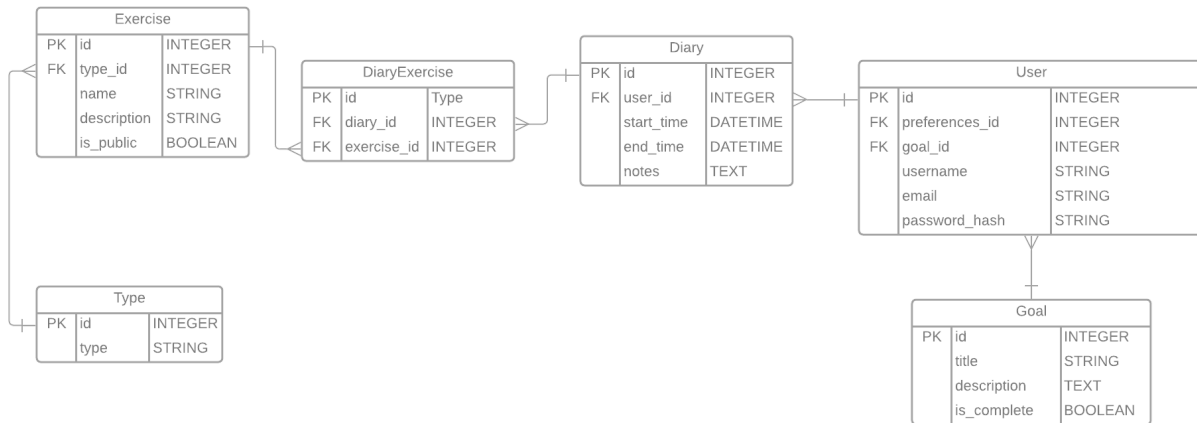
Database Notes and Layout.



This is the first iteration of the erd, preferences table will be used to save a set of user inputted data, i.e. ui preferences etc. Exercises that the user performs during that session can be saved and stored for future use and CAN (not must) also be made public. Description in the diary is used for noting down anything the user wants to note down for this specific session.



Changed some of the row names so they make more sense.



Added goal setting table, removed preferences too hard to store user preferences

routes* / function signatures** for each page

Landing // `@app.route('/') // def landing():`

Dashboard // `@app.route('/dashboard') // def dashboard():`

Diary // `@app.route('/diary') // def diary():`

Diary Log // `@app.route('/diary/log/<int:id>') // def diary_log(id):`

New Entry // `@app.route('/diary/add') // def diary_add(id):`

Edit Entry // `@app.route('/diary/edit/<int:id>') // def diary_edit(id):`

Goals // `@app.route('/goals') // def goals(id):`

Set Goals // `@app.route('/goal/add') // def goal_add(id):`

Edit Goals // `@app.route('/goal/edit/<int:id>') // def goal_edit(id):`

Search // `@app.route('/search', methods=["POST", "GET"]) // def search():`

Register // `@app.route('/register') // def register():`

Login // `@app.route('/login') // def login():`

Logout // `@app.route('/logout') // def logout():`

Website layout and design

For a webapp like this I want to make it very simple, and it must be functional, I want it to be able to get all the data and parse it into a readable format so that the user can view it and see their progression.

Search Bar



Search Options

Exercise: type

On: date

Before: date

During: date

After: date

Recent Searches

Exercise: monkey doo doo

On: 18/03/2022

Before: 18/03/2022

After: 18/03/2022

Notes: default search
searches by notes made
for each entry

Practice Tool

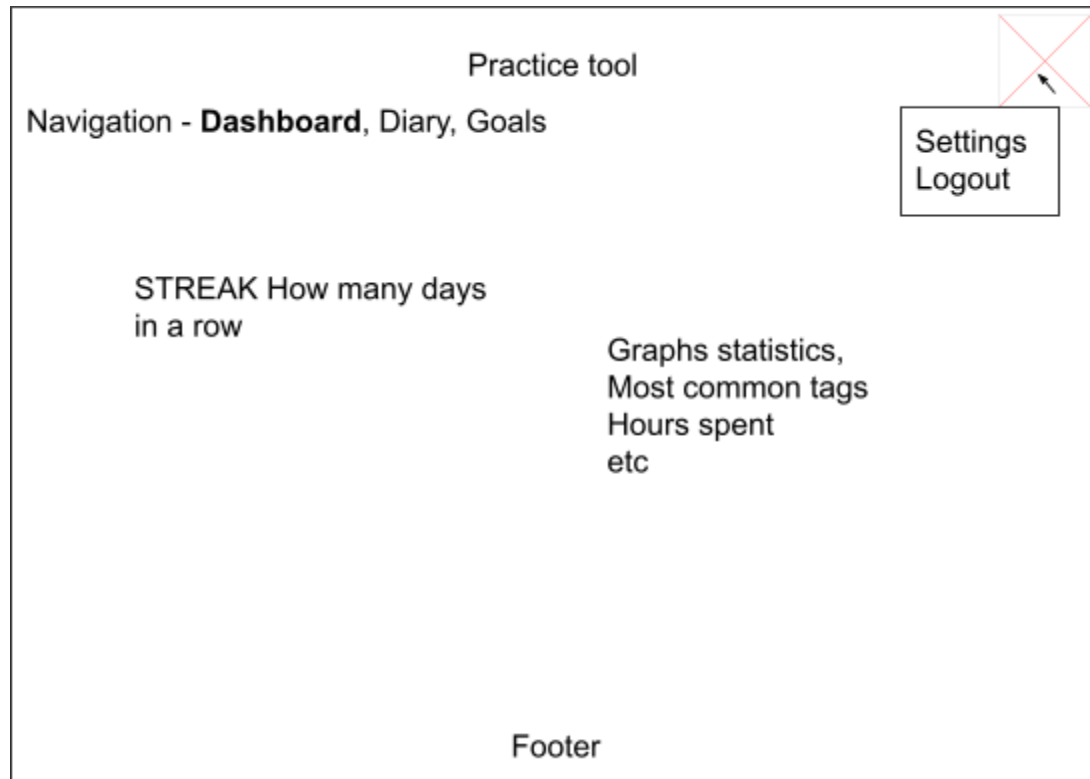
Login

password

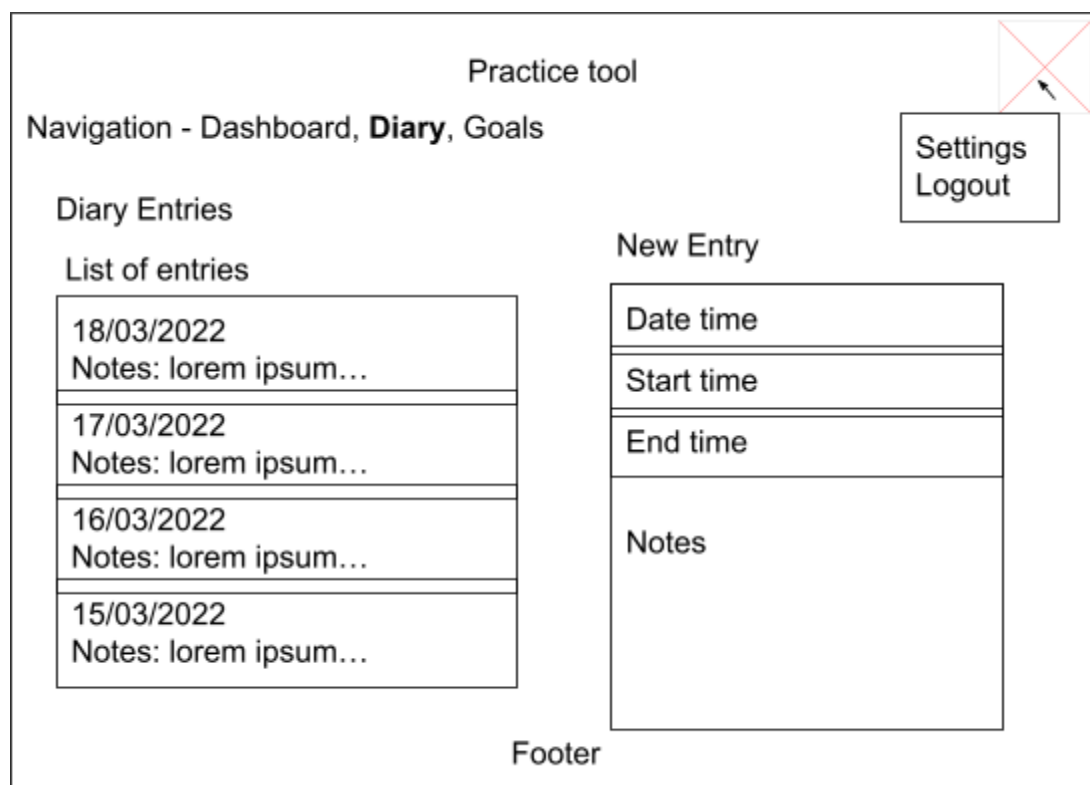
Forgot password

register

Login



On
hover
User
menu
appears



On
hover
User
menu
appears

Practice tool



Navigation - Dashboard, **Diary**, Goals

Settings
Logout

Diary Entries

Edit entry

List of entries

18/03/2022 Notes: lorem ipsum...
17/03/2022 Notes: lorem ipsum...
16/03/2022 Notes: lorem ipsum...
15/03/2022 Notes: lorem ipsum...

Date time
Start time
End time
Notes

Footer

Practice tool



Navigation - Dashboard, Diary, **Goals**

Settings
Logout

Goal Setting

Goals

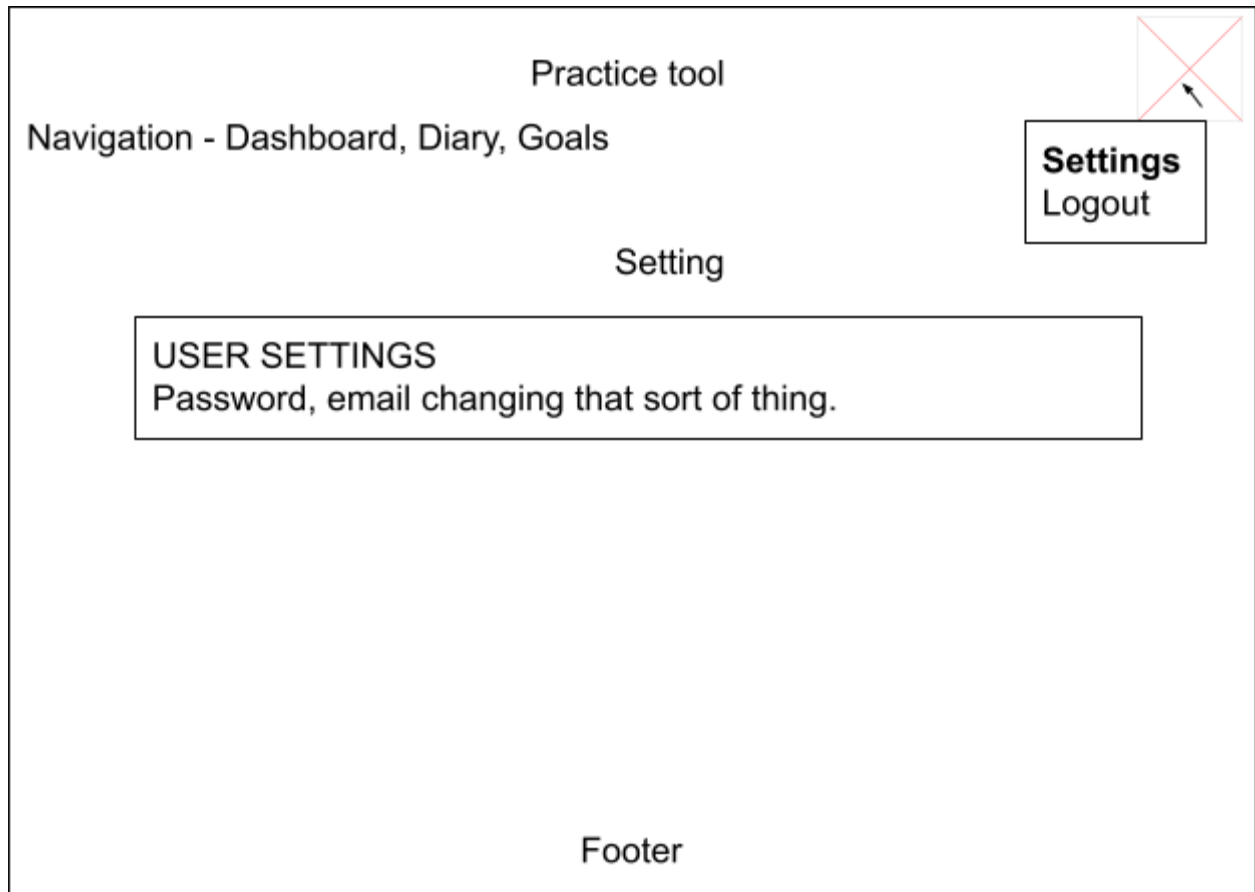
18/03/2022 Notes: lorem ipsum...
17/03/2022 Notes: lorem ipsum...
16/03/2022 Notes: lorem ipsum...
15/03/2022 Notes: lorem ipsum...

Goal Title

Whether you've achieved the goal or not

Description of the goal if the title is not enough

Footer



Database queries for each route (SQL query/queries) and expected sample results

query	Expected result	Purpose
<code>models.User.query.filter_by(username=form.username.data).first()</code>	Retrieve the first entry that matches user input	Used to get check if user inputted user is a user in the database
<code>models.User.query.get(int(id))</code>	Retrieve current user id	Get the id of the user logged in on current session
<code>models.User(email=form.email.data, username=form.email.data)</code>	Username and email added to user table	Used to register an account on the website.
<code>models.Diary.query.filter_by(user_id=current_user.id).all()</code>	Get all the diary entries for the current user.	Used to render all the user's diary entries

<code>models.Goal.query.filter_by(user_id=current_user.id).all()</code>	Get all the goal entries for the current user.	Used to render all the user's goals
<code>db.session.add(models.Goal(title=form.title.data, description=form.description.data, is_complete=form.is_complete.data))</code>	Adds user goals from user input with forms	So the user can input goals
<code>db.session.add(models.Diary(start_time=form.start_time.data, end_time=form.end_time.data, notes=form.notes.data))</code>	Adds user diary entries from user input with forms	So the user can use the diary

routes* / function signatures** for each page

```
@app.route('/', methods=['GET', 'POST'])
```

```
@login_required
```

```
def index():
```

```
@app.route('/diary/<int:id>', methods=['GET', 'POST'])
```

```
@login_required
```

```
def diary(id):
```

```
@app.route('/diary/create', methods=['GET', 'POST'])
```

```
@login_required
```

```
def create_diary():
```

```
@app.route('/diary/edit/<int:id>', methods=['GET', 'POST'])
```

```
@login_required
```

```
def edit_diary(id):
```

```
@app.route('/diary/delete/<int:id>')
```

```
@login_required
```

```
def delete_diary(id):
```

```
@app.route('/entry/<int:id>', methods=['GET', 'POST'])
```

```
@login_required
```

```
def entry(id):
```

```
@app.route('/entry/create/<int:id>', methods=['GET', 'POST'])
```

```
@login_required
```

```
def create_entry(id):
```



```
@app.route('/entry/delete/<int:id>')
@login_required
def delete_entry(id):
```

```
@app.route('/entry/<int:id>/tag/add/', methods=['GET', 'POST'])
@login_required
def add_tag(id):
```

```
@app.route('/entry/<int:id>/tag/remove/<int:tid>', methods=['GET', 'POST'])
@login_required
def remove_tag(id, tid):
```

```
@app.route('/login', methods=['GET', 'POST'])
def login():
```

```
@app.route("/register", methods=['GET', 'POST'])
def register():
```

```
@app.route("/logout")
@login_required
def logout():
```

Fonts / colours used

Fonts: Comic Sans MS

Colours: Off-white so it isn't as hard on the eyes and dark text that's in high contrast to the background so it is very legible, maybe textured to mimic paper

TEXT BLACK	Background #F3F3F3
BUTTON COLOUR #DDD	DANGER BUTTON #f44336

Tools used

IDE -Visual Studio Code, Atom

Database - Sqlalchemy

Languages - CSS, HTML5, Python Flask Library

General - Google Chrome, Github, lucid charts

Git - version control

Database Query Testing

What you are testing	How you are testing it	Expected results	Actual Results	Pass/Fail/Notes	Purpose
<code>models.Diary.query.filter_by(user_id=current_user.id).all()</code>	Running the code.	diary.title	Returns expected results	Pass	Query all of the user's diaries
<code>models.Diary.query.filter_by(user_id=current_user.id, id=id).all()</code>	Running the code.	diary.title diary.entry.date diary.entry.tags	Returns expected results	Pass	Query all of the entries of the user's specific diary.
<code>models.Diary(user_id=current_user.id, title=form.title.data)</code>	Running the code.	Adds user inputted form data to the database.	Returns expected results	Pass	Query to add a new diary to the database.
<code>models.Diary.query.filter_by(user_id=current_user.id, id=id).first_or_404()</code>	Running the code.	diary.title	Returns expected results	Pass	Retrieve a specific diary to delete or edit.
<code>models.Entry.query.filter_by(user_id=current_user.id, id=id).first_or_404()</code>	Running the code.	entry.diary.title entry.date entry.notes entry.tags entry.notes	Returns expected results	Pass	Get the data from a user's entry.
<code>models.Entry(user_id=current_user.id, diary_id=id, date=datetime.today().date())</code>	Running the code.	Inserts an entry where user id is current user's, and diary is the diary that the user is currently in, for today.	Returns expected results	Pass	Create a new entry in a specified diary for today.
<code>models.Tag.query.filter_by(user_id=current_user.id,</code>	Running the code.	tag.user_id tag.name	Returns expected results	Pass	Query user entered tags to see

name=name).first()					if it exists
models.Tag(user_id=current_user.id, name=name)	Running the code.	Creates row for user entered tag.	Returns expected results	Pass	Creating a tag.
models.User.query.filter_by(email=form.email.data).first()	Running the code.	user.id user.email user.password_hash	Returns expected results	Pass	Check if the user is registered.
models.User(email=form.email.data)	Running the code.	Creates a row for a new user.	Returns expected results	Pass	Registering user
user.set_password(form.password.data)	Running the code.	Sets a hashed password for the user.	Returns expected results	Pass	Set and encrypt the password.
user.check_password(form.password.data)	Running the code.	Check if user inputted the correct password.	Returns expected results	Pass	Verify that the password the user entered is correct.

Data Integrity and Testing.

User data integrity and testing

SOURCE: User input, all data comes from user input.

Register

test@test.test

.....

.....

Register

Already have an account?

[Click here to Login](#)

Structure

Data

Constraints

Indexes

Triggers

DDL

Grid view

Form view

+

-

✓

✕

⏪

⏩

1

⏪

⏩

🖨

🔗

🔗

🔍

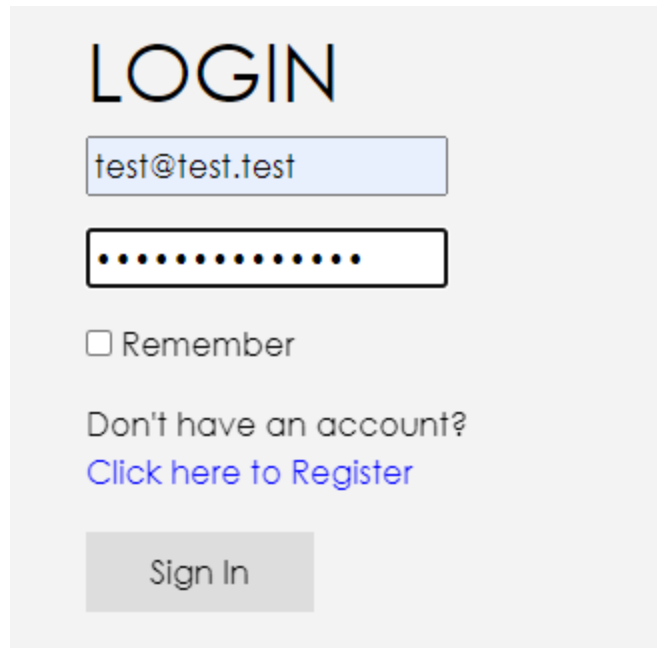
Filter data

📶

Total rows loaded: 2

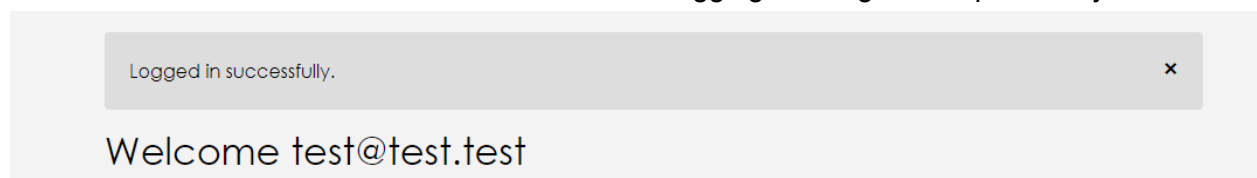
	id	email	password hash
1	1	18300@burnside.school.nz	pbkdf2:sha256:150000\$pwY6k0RS\$ec043511fe88fab14b364931ae5f74119a23dc46df238640bdf32533088fbcf
2	10	test@test.test	pbkdf2:sha256:260000\$FDUVNjryT23XXo9J\$ba1864ee8f7d1bb64656e3ecea7a97a170a68d1faad7d91bb09de88a9...

Display



A login form with a light gray background. At the top is the word "LOGIN" in large, black, sans-serif capital letters. Below it is a text input field containing "test@test.test". Underneath that is a password input field represented by a row of 12 black dots. Below the password field is a checkbox labeled "Remember". Further down is the text "Don't have an account?" followed by a blue link "Click here to Register". At the bottom is a gray button with the text "Sign In".

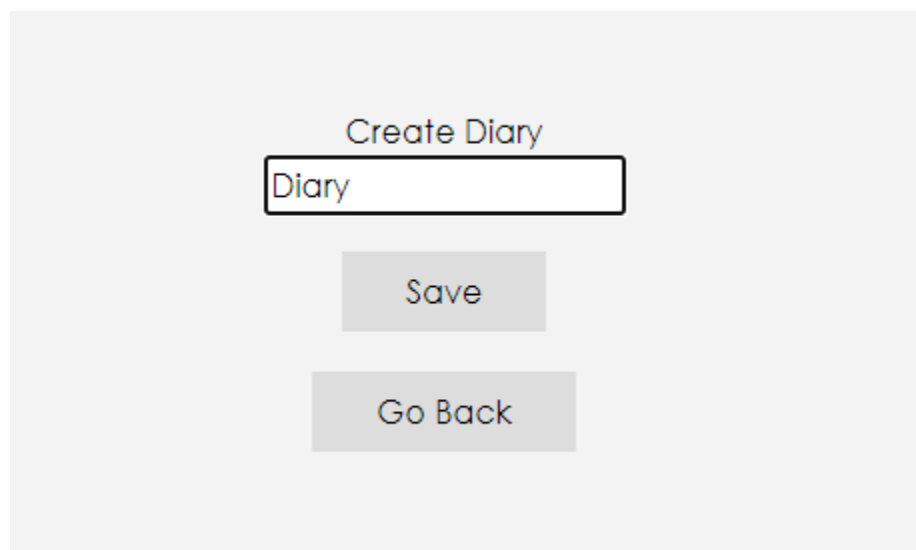
logging in using details previously entered



A confirmation screen with a light gray background. At the top is a gray notification bar with the text "Logged in successfully." and a close button (an 'x' in a square). Below the bar is the text "Welcome test@test.test" in a black, sans-serif font.

Logged in successfully current user email matches entered email therefore data is correctly displayed and integrouous. For user objects.

Diary data integrity and testing



A form for creating a diary entry on a light gray background. At the top is the text "Create Diary" in a blue, sans-serif font. Below it is a text input field containing the word "Diary". Underneath the input field are two gray buttons: "Save" and "Go Back", stacked vertically.

Inputted title for diary as "Diary"

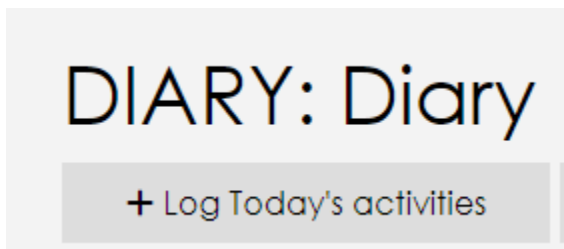
	id	user id	title
1	33	10	Diary

The test user's id is 10 so this is correct, title is correctly displayed in the database as well.



Diary has correct name, hovering over link reveals `127.0.0.1:5000/diary/33` correct id for the diary.

Entry data integrity and testing.



clicked log today's activities

Created new entry for "2022-09-27" ×

[→ Diary](#) [→ September 27, 2022](#)

September 27, 2022

+ Add Tags × Delete This Entry

B *I* | ?

Save

Created a new entry.

	id	user id	diary id	date	notes
1	26	10	33	2022-09-27	NULL

Date and user are correct and so is diary id. `127.0.0.1:5000/entry/26` The entry number is also correct.

Saved notes ×

→ [Diary](#) → [September 27, 2022](#)

September 27, 2022

+ Add Tags

× Delete This Entry

B *I* | ?

This is a diary entry

Save

	id	user id	diary id	date	notes
1	26	10	33	2022-09-27	<p>This is a diary entry</p>

Notes update when I save notes. Data is correctly displayed on the website.

→ [Diary](#) → [September 27, 2022](#)

September 27, 2022

[+ Add Tags](#)[× Delete This Entry](#)

B *I* | | | | ?

Field cannot be longer than 2000 characters.

[Save](#)

Checking what happens when more than 2000 characters are added and the user tries to save.

→ [Diary](#) → [September 27, 2022](#)

September 27, 2022

[+ Add Tags](#)[× Delete This Entry](#)

B *I* | | | | ?

This field is required.

[Save](#)

Testing what happens when nothing is entered into notes and the user tries to save.

Adding tags data integrity and testing

Source:



Add a Tag

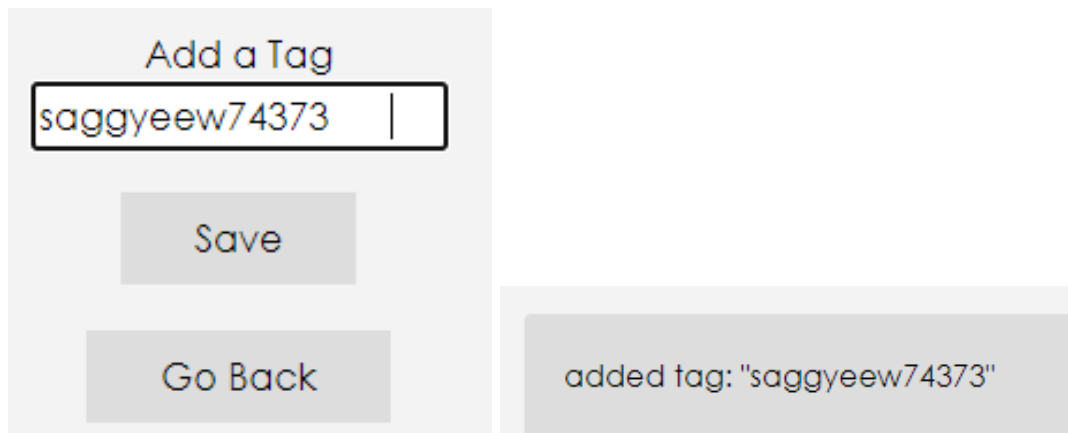
Tag

Save

Go Back

Tag is supposed to be formatted to remove all capitalisation and trailing spaces.

Testing trailing spaces.



Add a Tag

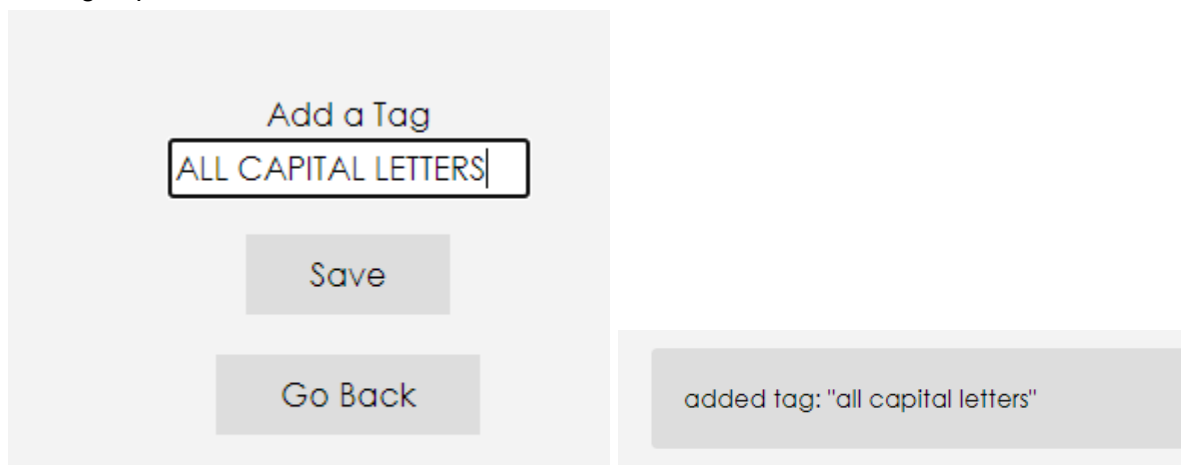
saggyeew74373

Save

Go Back

added tag: "saggyeew74373"

Testing capital letters



Add a Tag

ALL CAPITAL LETTERS

Save

Go Back

added tag: "all capital letters"

Testing if field is longer than the character limit (36 characters)

Add a Tag

safafaasfassafafaasfa:

Field cannot be longer than 36 characters.

Save

Go Back

	id	user id	name
1	17	10	tag

correct user.

Tags:

tag ✕

name is correct.

127.0.0.1:5000/entry/26/tag/remove/17

hovering over the function to remove tag shows the correct tag id, therefore the data entered is correct.

Testing navigation buttons

→ [Diary](#) → [September 27, 2022](#)

[→ Diary](#) [→ September 27, 2022](#)

September 27, 2022

+ Add Tags

× Delete This Entry

B *I* |   |   |   | ?


Save

Pressing september 27, 2022 takes me back to this page.

[→ Diary](#) [→ September 27, 2022](#)

DIARY: Diary

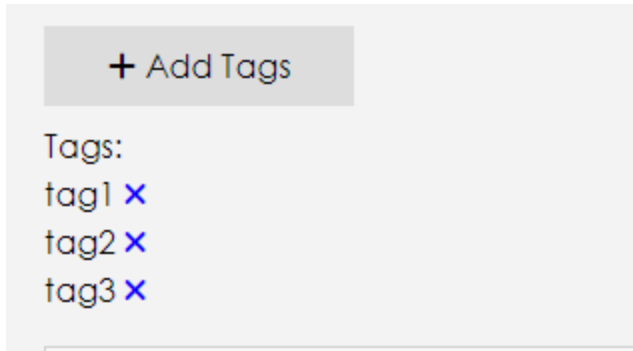
+ Log Today's activities

 Edit Title

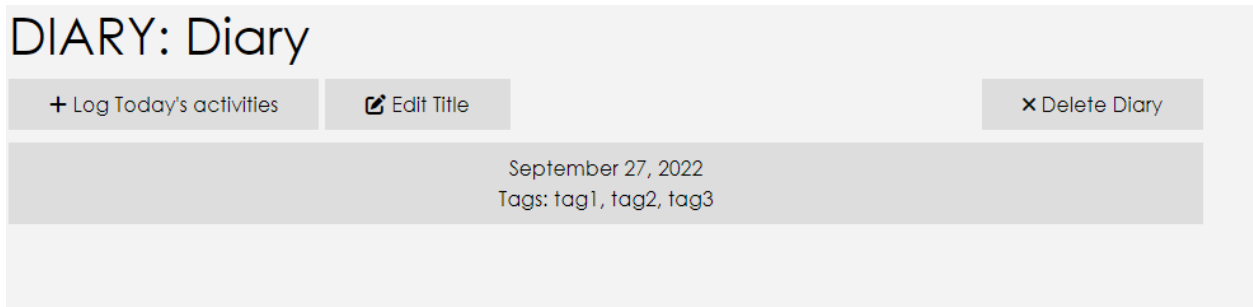
× Delete Diary

September 27, 2022

Pressing the diary takes me back to the diary.



Checking tags are displayed properly when added.

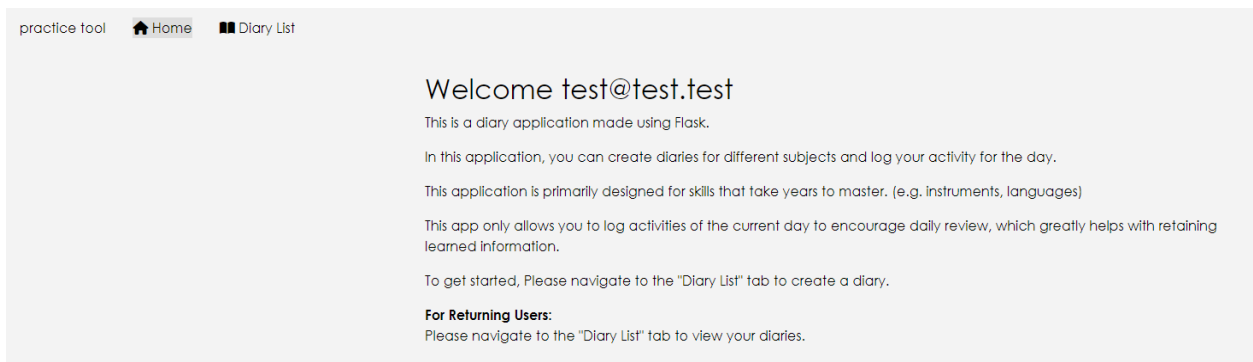


checking that there are no trailing commas on the main entries page.

Testing

Testing routes.

Home button takes me to the home page.



Diary list page takes me to diary list page.



Login button takes me to login page

practice tool Login

LOGIN

Email

Password

☐ Remember

Don't have an account?
[Click here to Register](#)

Register button takes me to register page

practice tool Login Register

Register

Email

Password

Confirm Password

Already have an account?
[Click here to Login](#)

Testing login functions

When invalid email is entered

wrong password or email ×

LOGIN

Email

Password

☐ Remember

Don't have an account?
[Click here to Register](#)

Testing when invalid email address is entered into the register, and also when password confirmation does not match previously entered password.

Register

Invalid email address.

Field must be equal to password.

Already have an account?
[Click here to Login](#)

Testing what happens when inputs have more characters than the character limit. (256, 36 respectively)

Register

Invalid email address. Field cannot be longer than 256 characters.

Field cannot be longer than 36 characters.

Field cannot be longer than 36 characters.

Testing when a user tries to register with an existing user's email address.

User already exists ×

Register

Already have an account?
[Click here to Login](#)

When a user enters a valid email and passwords that match. Redirects and informs them.

You are now a registered user.

×

LOGIN

Email

Password

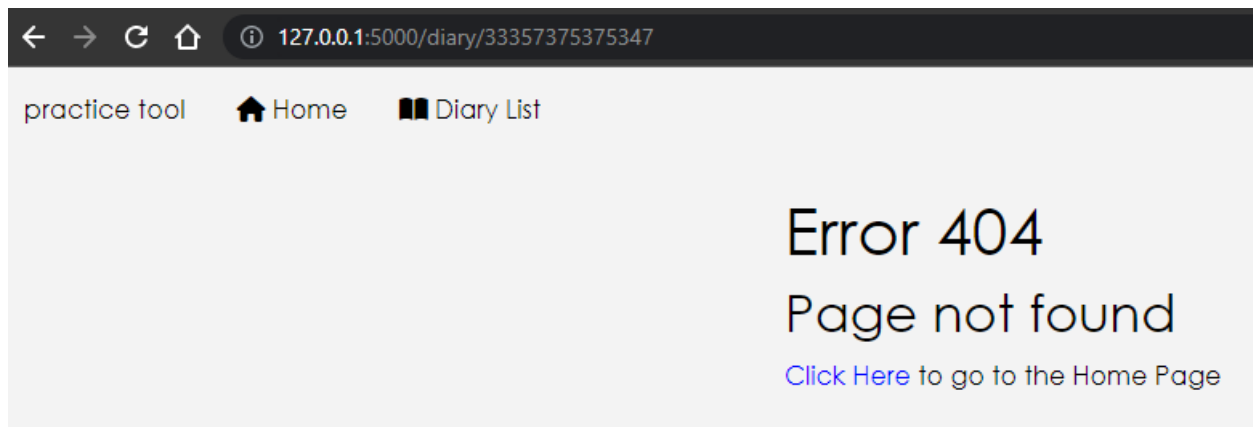
☐ Remember

Don't have an account?
[Click here to Register](#)

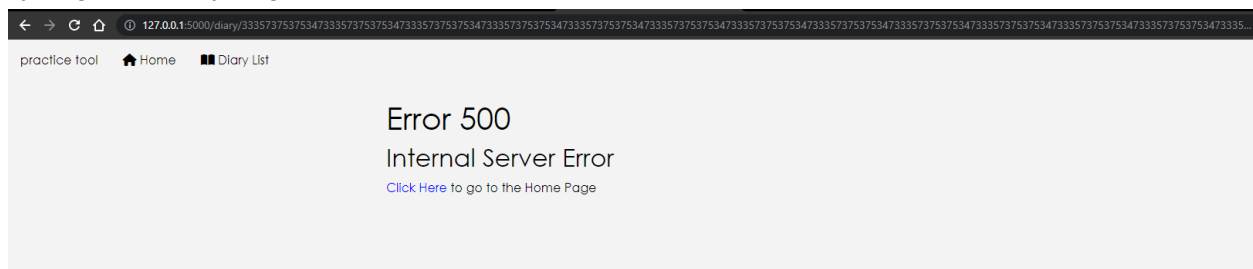
Sign In

Testing to see typing url bar would break it.

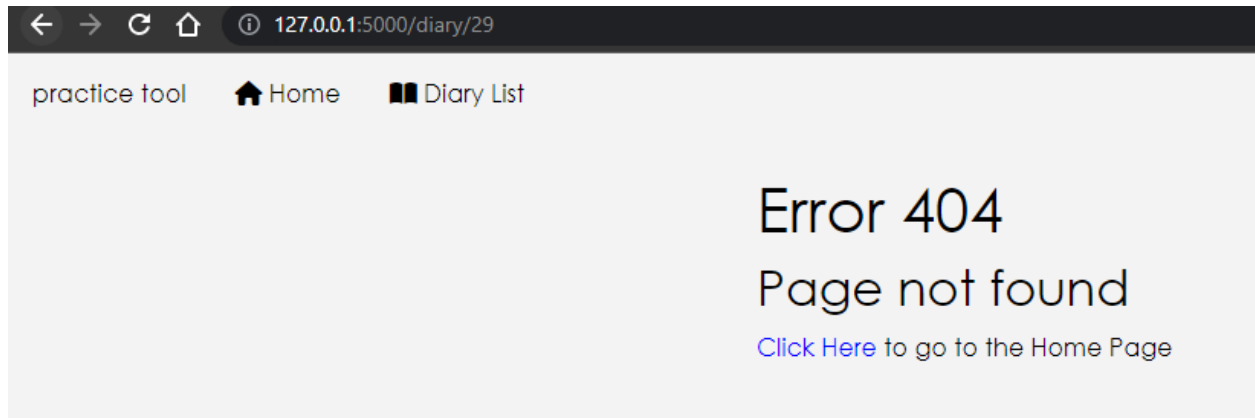
Typing random numbers



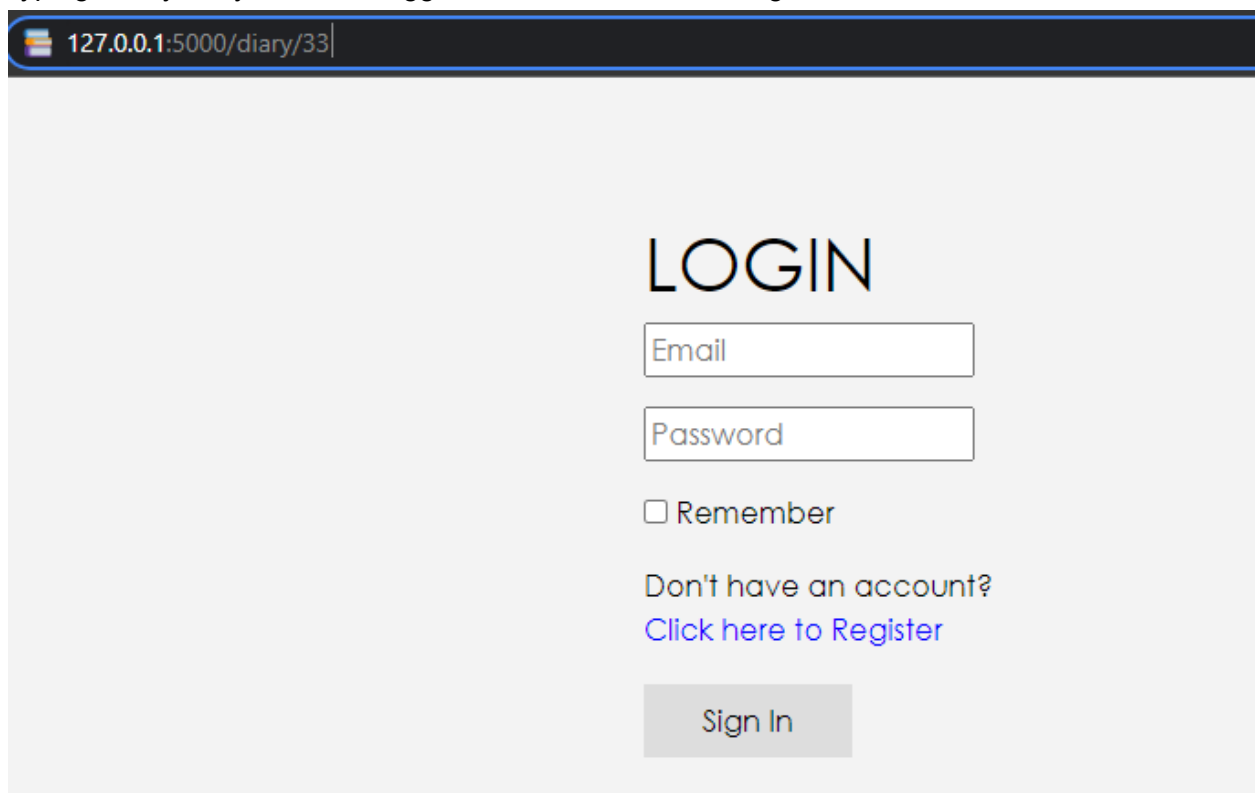
Typing extremely large number



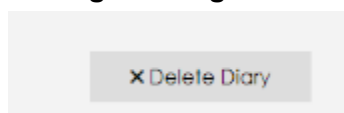
Typing in a diary owned by another user.



Typing in any diary while not logged in sends me back to login.



Testing deleting



DELETEING DIARY: "Diary"

Are you sure you want to delete this diary?
This diary cannot be recovered once deleted.

Delete

Go Back

Diary: "Diary" was deleted

x

+ Create



id

user id

title

"diary" no longer exists.

Testing entry delete

✕ Delete This Entry

DELETEING ENTRY FOR "September 27, 2022"

Are you sure you want to delete this entry?
This entry cannot be recovered once deleted.

Delete

Go Back

Deleted entry for "September 27, 2022"

✕

DIARY: Diary

+ Log Today's activities

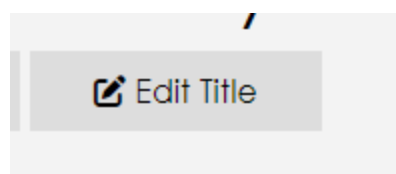
✎ Edit Title

✕ Delete Diary

id	user id	diary id	date	notes	

entry no longer exists.

Editing diary title.



Edit Title

Edited Diary Title

Save

Go Back

Diary title has been changed to: "Edited Diary Title" x

DIARY: Edited Diary Title

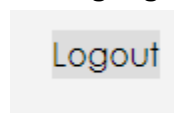
+ Log Today's activities Edit Title x Delete Diary

	id	user id	title
1	34	10	Diary

	id	user id	title
1	34	10	Edited Diary Title

id 34 has been changed from title has been changed from "diary" to "edited diary title"

Testing logout



Route takes me back to the login screen.

You have successfully logged out. ×

LOGIN

Email

Password

☐ Remember

Don't have an account?
[Click here to Register](#)

Sign In

Testing what happens when I add an existing tag to an entry.

"a" has already been used in this entry ×

[→ Diary](#) [→ September 25, 2022](#)

September 25, 2022

+ Add Tags

× Delete This Entry

Tags:
12512312 ×
a ×
gasgasg ×
gasgsag ×

Tells the user the tag is already associated with the entry.

Testing when nothing is entered into the title for creating a diary.

Create Diary

Title

!

Please fill out this field.

Go Back

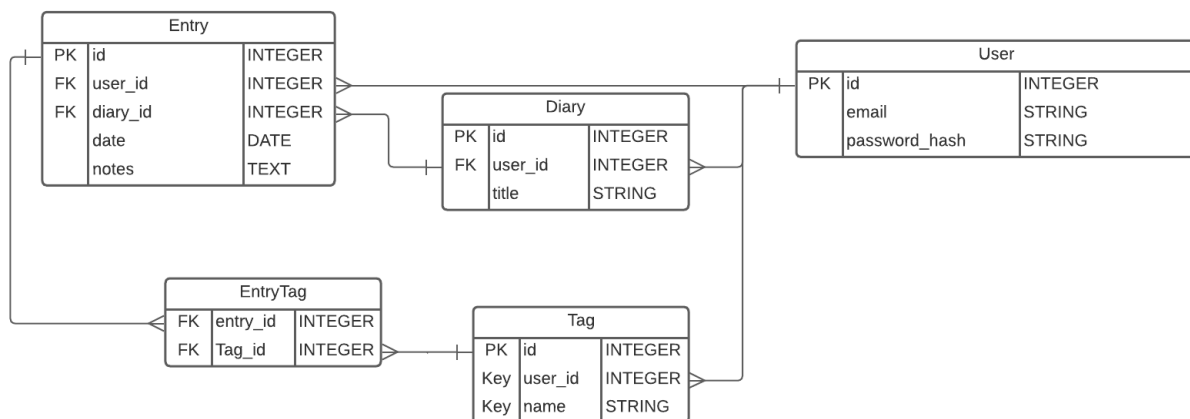
tells the user to fill out the field. passes the testing.

Testing when nothing is entered into tag adding box



tells user to fill out the field, passes testing.

Database changes due to testing



Removed Exercises because it would be impossible to manage a database where users can interact with other user's exercises, malicious code could be injected into the exercise which could cause someone to lose control of their computer.

Goal setting was not manageable in the time frame set for myself.

Added tags so that it would be easier to see basic information of what you did during that day without clicking on the entry.

Removed activity table because, adding times require user to enter times would be difficult to make sure that one end time > start time.

Website changes due to testing.

All pretty true to the design. However, as having everything on one page would require javascript I decided to split everything off to different pages. It is also best practice to only have one form per page, as the program cannot differentiate between the forms well.

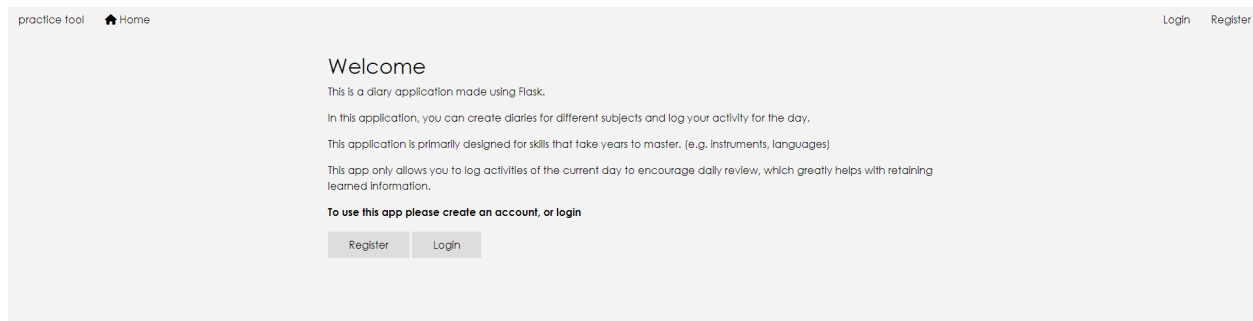
I removed the dropdown to manage user settings and logout, as there is not really much to configure with this website. And having a dropdown for just logout is adding complexity for no reason.

Dashboard has been replaced with an index page telling the user what the website is used for, dashboard would require.

Mr Dunford's comments

“User greeting? Dumped straight to a login page with no information about the site or what its for... secret society?”

I added a landing page so that the user knows what the website is for and how to use it.

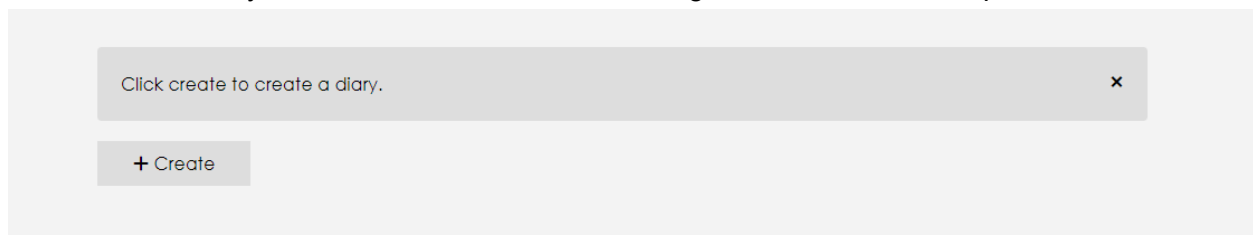


“Tell everyone to go to the diary list - why not just send them there.”

The user is now redirected to the diary list instead of the home page.

“Not immediately apparent I should click on bob or whatever diary?”

I added a flash message to tell the user to click the title of the diary to open it. When the user has not created any diaries I've added a flash message to tell them what to press to create one.



multiple same tag adds possible - ???

Added a flash message to show the user that the tag is already associated with the entry that they are trying to add to.



Relevant Implications

Database Relevant Implications

Intellectual property - is a category of property which includes intangible creations, copyrights, trademarks, patents, etc. Other people will not be able to use your intellectual property unless given explicit permission by the creator. Creations of the mind, as the primary use of a website is to record what you did during the day, it is reality and not a creation of the mind. I am not sourcing any text or images from anywhere else therefore my website does not infringe on the copyrights of any creators.

Sustainability and Future Proofing - refers to whether the database is designed so that it will remain functional in the future. In this case, my database is designed so that it can adapt to changes in my project. My database has different tables for entries and diaries so they can be dynamically added to the database without slowing down the system. By organising data into a related table; we eliminate redundancy and increase the integrity which improves performance of the query.

Privacy - My website does not store any personal information besides email, and notes. (Tags could possibly be considered personal information" This information is password protected and hashed so they wouldn't be able to see the information without logging in first.

Web relevant implications

Aesthetics - refers to how the website looks to the user. It may include things like layout, colour scheme and fonts. Maintaining good aesthetics is important because it can create good impressions for the user and increase the effectiveness of the website. My website addresses

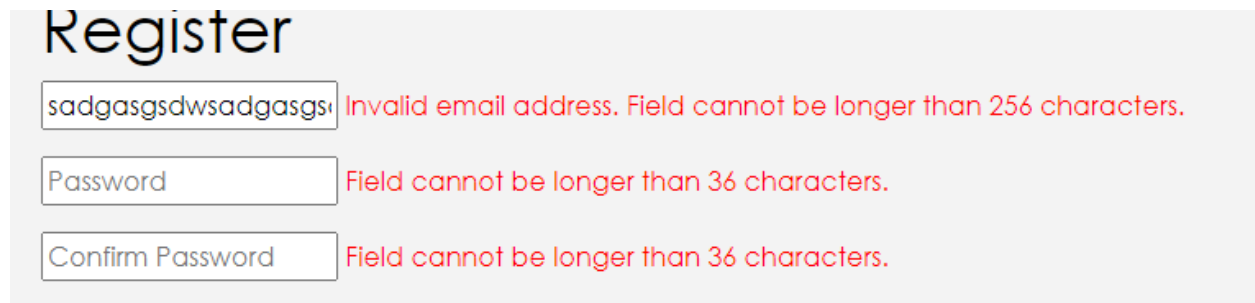
this by using minimalistic and simplistic colours and fonts that are easy for the user to read. It also uses a logical layout and attracts the user's attention to the more crucial information in the centre while leaving the not so crucial information such as navigation and searching stuff, on the top. The navigation bar on the side, main content next to it, header at the top and footer at the bottom makes it easy for the user to navigate through the pages with colours and fonts that are easy to read from.

Usability - refers to the user's experience involving consistency, error prevention and efficiency. Having a usable website is important as keeping things consistent and preventing errors is key to keeping a good user's experience. My website addresses this by having features to prevent errors e.g when a user does not properly fill out a field in my search bar, a notice will come up warning them that what they have put in is not suitable, this prevents an error from occurring. I have also kept my website consistent, whenever you hover over something clickable, it will make the background darker showing that it's a link to a new page.

Relevant conventions

Usability Heuristics

Help users recognize, diagnose, and recover from errors - I have warnings put in place to warn users when input by the user does not reach the requirements. E.g when a user enters something with more than 36 characters, it will display a message asking for less than 36 characters.



Register

Invalid email address. Field cannot be longer than 256 characters.

Field cannot be longer than 36 characters.

Field cannot be longer than 36 characters.

Match between system and the real world - Diaries are laid out in a list ordered by when they were made, which is an accepted sorting method. Entries are sorted in descending order with most recent at the top and oldest at the bottom, this is common practice among other diary apps.

User control and freedom - have easy to access exits when pop ups show up, so the user doesn't feel trapped and feels like that they are in control. My website does this by having easy and clear exits.



Consistency and standards - making sure that the user should not have to wonder what each thing does, and force them to learn something new. When the user hovers over the thread the

Visibility of system status - the design should keep the user informed when something is happening. In my website this is addressed when a user hovers over a link to a new page or a item on the main part of the page, the item is then greyed out and the user knows that something might happen if they click on it.

Sprints

Sprint 1

Goal: Create my Register page and Login page for my website and make sure it works.
Creating models. Make sure navigation to other pages works.

What	Code	Results	Pass/ Fail
Creates a register page with working forms that redirects to the login page when correct information is entered.	<pre>+ @app.route("/register", methods=['GET', 'POST']) + def register(): + if current_user.is_authenticated: + return redirect(url_for('index')) + form = RegistrationForm() + if form.validate_on_submit(): + user = models.User(email=form.email.data, username=form.email.data) + user.set_password(form.password.data) + db.session.add(user) + db.session.commit() + flash('You are now a registered user.') + return redirect(url_for('login')) + return render_template("register.html", form=form) + class RegistrationForm(FlaskForm): + username = StringField('Username', validators=[DataRequired(), Length(max=256)], render_kw={"placeholder": "Username"}) + email = StringField('Email', validators=[DataRequired(), Email(), Length(max=256)], render_kw={"placeholder": "email"}) + password = PasswordField('Password', validators=[DataRequired(), Length(max=36)], render_kw={"placeholder": "Password"}) + # makes sure 2nd password is same as 1st password + password2 = PasswordField(+ 'Repeat Password', validators=[DataRequired(), EqualTo('password')], Length(max=36), render_kw={"placeholder": "Confirm Password"}) + submit = SubmitField('Register')</pre>	<p>Register</p> <p>Email <input type="text"/></p> <p>Password <input type="password"/></p> <p>Confirm Password <input type="password"/></p> <p><input type="button" value="Register"/></p> <p>Already have an account? Click here to Login</p>	Pass

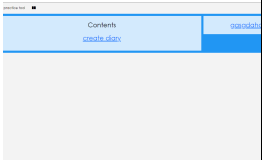
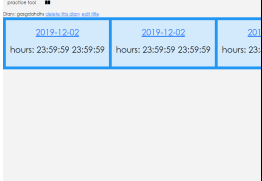
	<pre> ... @@ -0,0 +1,16 @@ 1 + {% if form.username.errors %} 2 + <ul class="errors"> 3 + {% for error in form.username.errors %} 4 + {{ error }} 5 + {% endfor %} 6 + 7 + {% endif %} 8 + 9 + <form class="register" method="post"> 10 + {{ form.csrf_token }} 11 + {{form.username}} 12 + {{form.email}} 13 + {{form.password}} 14 + {{form.password2}} 15 + <input type="submit" value="Go"> 16 + </form> </pre>		
Creates a login page with working forms that redirects to the index page when correct information is entered.	<pre> 29 + @app.route('/login', methods=['GET', 'POST']) 30 + def login(): 31 + if current_user.is_authenticated: 32 + return redirect(url_for('index')) 33 + form = LoginForm() 34 + if form.validate_on_submit(): 35 + user = models.User.query.filter_by(username=form.username.data).first() 36 + if user is None or not user.check_password(form.password.data): 37 + flash('wrong password or username') 38 + else: 39 + login_user(user, remember=form.remember_me.data) 40 + flash('Logged in successfully.') 41 + next = request.args.get('next') 42 + if not is_safe_url(next, allowed_hosts="localhost:5000"): 43 + return abort(400) 44 + return redirect(next or url_for('index')) 45 + return render_template('login.html', form=form) </pre> <p>templates/login.html</p> <pre> ... @@ -0,0 +1,7 @@ 1 + <form class="login" method="post"> 2 + {{ form.csrf_token }} 3 + {{form.username()}} 4 + {{form.password()}} 5 + {{form.remember_me()}} 6 + <input type="submit" value="Go"> 7 + </form> </pre> <pre> + class LoginForm(FlaskForm): + username = StringField('Username', validators=[DataRequired(), Length(max=256)], render_kw={"placeholder": "Username"}) + password = PasswordField('Password', validators=[DataRequired(), Length(max=36)], render_kw={"placeholder": "Password"}) + remember_me = BooleanField('Remember Me') + submit = SubmitField('Sign In') </pre>	<h2>LOGIN</h2> <p>Email <input type="text"/></p> <p>Password <input type="password"/></p> <p><input type="checkbox"/> Remember</p> <p>Don't have an account? Click here to Register</p> <p><input type="submit" value="Sign In"/></p>	Pass
Test to see whether user is logged in when redirecting to the index page	<pre> + @login_manager.user_loader + def load_user(id): + return models.User.query.get(int(id)) + + + @app.route('/', methods=['GET']) + @login_required + def index(): + return render_template('index.html') </pre>	<p>Hi <User<test@test.test>!</p>	Pass

	<pre> ... @@ -0,0 +1,7 @@ 1 + {% extends 'layout.html' %} {% block content %} 2 + {% if current_user.is_authenticated %} 3 + Hi {{ current_user.username }}! 4 + {% endif %} 5 + 6 + 7 + {% endblock %} </pre>		
Creating models	<pre> 56 lines (47 sloc) 2.14 KB 1 from unicodedata import name 2 from sqlalchemy import ForeignKey 3 from app import db 4 from werkzeug.security import generate_password_hash, check_password_hash 5 from flask_login import UserMixin 6 from datetime import datetime 7 8 class User(UserMixin, db.Model): 9 __tablename__ = 'User' 10 id = db.Column(db.Integer, primary_key=True, nullable=False) 11 preferences_id = db.Column(db.String, ForeignKey('Preferences.id')) 12 username = db.Column(db.Text, nullable=False) 13 email = db.Column(db.Text, nullable=False, unique=True) 14 password_hash = db.Column(db.Text, nullable=False) 15 16 def set_password(self, password): 17 self.password_hash = generate_password_hash(password) 18 19 def check_password(self, password): 20 return check_password_hash(self.password_hash, password) 21 22 def __repr__(self): 23 return '<User{}>'.format(self.username) 24 25 class Type(db.Model): 26 __tablename__ = 'Type' 27 id = db.Column(db.Integer, primary_key=True, nullable=False) 28 type = db.Column(db.String, primary_key=True, nullable=False) 29 30 class Preference(db.Model): 31 __tablename__ = 'Preference' 32 id = db.Column(db.Integer, primary_key=True, nullable=False) 33 preference = db.Column(db.String, nullable=False) 34 35 class Exercise(db.Model): 36 __tablename__ = 'Exercise' 37 id = db.Column(db.Integer, primary_key=True, nullable=False) 38 type_id = db.Column(db.Integer, nullable=False) 39 name = db.Column(db.String, nullable=False) 40 description = db.Column(db.String, nullable=False) 41 is_public = db.Column(db.Boolean, nullable=False) 42 43 class Diary(db.Model): 44 __tablename__ = 'Diary' 45 id = db.Column(db.Integer, primary_key=True, nullable=False) 46 user_id = db.Column(db.Integer, ForeignKey('User.id'), nullable=False) 47 date = db.Column(db.Date, nullable=False) 48 start_time = db.Column(db.Time, nullable=False) 49 end_time = db.Column(db.Time, nullable=False) 50 notes = db.Column(db.Text) 51 52 class DiaryExercise(db.Model): 53 __tablename__ = 'DiaryExercise' 54 id = db.Column(db.Integer, primary_key=True, nullable=False) 55 diary_id = db.Column(db.Integer, ForeignKey('Diary.id')) 56 exercise_id = db.Column(db.Integer, ForeignKey('Exercise.id')) </pre>		Pass

Checking to see if links show up and work.	<pre> 7 lines (7 sloc) 156 Bytes 1 <!-- The topnav --> 2 <div class="topnav"> 3 4 Dashboard 5 Diary 6 7 </div> </pre>	<ul style="list-style-type: none"> • Dashboard • Diary 	Pass
--	--	--	------

Sprint 2

Goal: Get text area and make diaries, entries to work and display the data correctly.

What	Code	Results	Pass/Fail
Making diaries display on the page	<pre> @app.route('/diary/<int:id>', methods=['GET', 'POST']) @login_required def diary(id): diary = models.Diary.query.filter_by(user_id=current_user.id, id=id).all() return render_template('diary.html', diary=diary) </pre> <pre> {% extends 'layout.html' %} {% block content %} <div id="wrapper"> <div class="grid-container"> <div class="item1"> <p>Contents</p> <p>create diary</p> </div> {% for diary in diary %} <div>{{diary.title}}</div> {% endfor %} </div> </div> {% endblock %} </pre>		Pass
Displaying entries with the times.	<pre> {% extends 'layout.html' %} {% block content %} {% for diary in diary %} Diary: {{diary.title}} delete this diary edit title <div class="grid-container"> {% for entry in diary.entries %} <div class="entry"> <p>{{entry.date}}</p> {% for activity in entry.activities %} <p>hours: {{activity.start_time}} {{activity.end_time}}</p> {% endfor %} </div> {% endfor %} </div> {% endblock %} </pre>		Pass

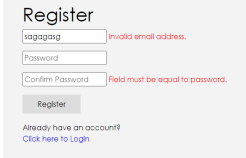
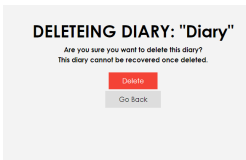

Displaying individual entries and their notes.	<pre> @app.route('/entry/<int:id>', methods=['GET', 'POST']) @login_required def entry(id): form = EntryForm() entry = models.Entry.query.filter_by(user_id=current_user.id, id=id).first_or_404() if form.validate_on_submit(): entry.notes = form.notes.data db.session.merge(entry) db.session.commit() form.notes.data = entry.notes return render_template('entry.html', entry=entry, form=form) </pre> <pre> {% extends 'layout.html' %} {% block content %} <div class="entry"> back <form class="entry" method="post"> {{ form.csrf_token }} <div class="grid-container"> <div class="entry_info"> <p>{{entry.date}}</p> {% for activity in entry.activities %} <p>hours of practice: {{activity.end_time}} {{ activity.start_time}}</p> {% endfor %} {{ form.start_time }} {{ form.end_time }} </div> <div class="text-editor"> {{ form.notes }} </div> </div> {{ form.submit }} </form> </div> {% endblock %} </pre>		Pass
Create diary	<pre> @app.route('/diary/create', methods=['GET', 'POST']) @login_required def create_diary(): form = DiaryForm() if form.validate_on_submit(): diary = models.Diary(user_id=current_user.id, title=form.title.data) db.session.add(diary) db.session.commit() flash('Created new Diary') return redirect(url_for('index')) return render_template('create_diary.html', form=form) </pre>		Fail won't add to the database.
Create diary	<pre> @app.route('/diary/create', methods=['GET', 'POST']) @login_required def create_diary(): form = DiaryForm() if form.validate_on_submit(): diary = models.Diary(user_id=current_user.id, title=form.title.data) db.session.add(diary) db.session.commit() flash('Created new Diary') return redirect(url_for('index')) return render_template('create_diary.html', form=form) </pre>		Pass merge makes it add to database.
Create entries	<pre> @app.route('/entry/<int:id>', methods=['GET', 'POST']) @login_required def create_entry(id): form = EntryForm() entry = models.Entry.query.filter_by(user_id=current_user.id, id=id).first_or_404() if form.validate_on_submit(): entry.notes = form.notes.data db.session.merge(entry) db.session.commit() form.notes.data = entry.notes return render_template('entry.html', entry=entry, form=form) </pre>		

Delete entries	<pre> app.route('/entry/delete/<int:id>') def delete_entry(id): entry_to_delete = models.Entry.query.filter_by(user_id=current_user.id, id=id).first() try: db.session.delete(entry_to_delete) db.session.commit() #return a message flash("entry post was deleted") return redirect(url_for('diary', id=entry_to_delete.diary.id)) except: flash("whoops there was a problem deleting this entry") return redirect(url_for('diary', id=entry_to_delete.diary.id)) </pre>		Pass
Edit diary title	<pre> @app.route('/diary/edit/<int:id>', methods=['GET', 'POST']) @login_required def edit_diary(id): form = DiaryForm() diary = models.Diary.query.filter_by(user_id=current_user.id, id=id).first_or_404() if form.validate_on_submit(): diary.title = form.title.data db.session.merge(diary) db.session.commit() flash("edited new diary name") return redirect(url_for('diary', id=diary.id)) return render_template('create_diary.html', form=form) </pre>		Pass
Delete diary	<pre> @app.route('/diary/delete/<int:id>') def delete_diary(id): diary = models.Diary.query.filter_by(user_id=current_user.id, id=id).first() local = db.session.merge(diary) db.session.delete(local) db.session.commit() #return a message flash("entry post was deleted") return redirect(url_for('index')) </pre>		Pass

Sprint 3

Goal: Work on error handling. Make Flash messages and HCI.


What	Code	Results	Pass/Fail
Testing flash message	<pre> <div class="column message"> <p>{% with messages = get_flashed_messages() %} {% if messages %} {% for message in messages %} <div class="alert"> &times; {{message}} </div> {% endfor %} {% endif %} {% endwith %} </pre>		Pass

<p>Making sure the it shows the errors with the form to the user.</p>	<pre> <h1>Register</h1> <form class="register" method="post"> {{form.csrf_token}} <p>{{form.email}} {% for error in form.email.errors %} {{ error }} {% endfor %} </p> <p>{{form.password}} {% for error in form.password.errors %} {{ error }} {% endfor %}</p> <p>{{form.password2}} {% for error in form.password2.errors %} {{ error }} {% endfor %}</p> <p>{{form.submit(class="button")}}</p> </form> <p>Already have an account?
 Click here to Login </p> {% endblock %} </pre>		<p>Pass</p>
<p>Adding pages confirming that they want to delete certain entries</p>	<pre> {% extends 'layout.html' %} {% block content %} <center> <h1>DELETING DIARY: "{{diary.title}}"</h1> <p>Are you sure you want to delete this diary?
 This diary cannot be recovered once deleted.</p> <form class="delete_diary" method="post"> {{ form.csrf_token }} {{ form.submit(class="button", style="background-color:#f44336; color:white;") }} </form> Go Back </center> {% endblock %} </pre>		<p>Pass</p>
<p>Making an index page showing what the purpose of the website is.</p>	<pre> {% extends 'layout.html' %} {% block content %} <h2>Welcome {{current_user.email}}</h2> <p>This is a diary application made using Flask.</p> <p>In this application, you can create diaries for different subjects and log your activity for the day.</p> <p>This application is primarily designed for skills that take years to master. (e.g. instruments, languages)</p> <p>This app only allows you to log activities of the current day to encourage daily review, which greatly helps with retaining learned information.</p> <p>To get started, Please navigate to the "Diary List" tab to create a diary.</p> <p>For Returning Users:
 Please navigate to the "Diary List" tab to view your diaries.</p> {% endblock %} </pre>		<p>Pass</p>

Iterative improvement

Commits on Mar 9, 2022


login framework
idoyt committed on Mar 9

 4882016 <>


On Mar 9, 2022 I created my login framework with flask, flask forms, and sqlalchemy.

Commits on Mar 14, 2022

styling
idoyt committed on Mar 14

 398af99 <>

models and some templating
idoyt committed on Mar 14

 bfb66a2 <>

On Mar 14, 2022 I did some styling of my pages to make them look better and added a navigation bar to test my routes.

Commits on Mar 28, 2022

search testing
idoyt committed on Mar 28

 09e42d7 <>

On Mar 28, 2022 I created a search function to see if I could search for keywords in my diary notes.

Commits on May 9, 2022


creating diary and displaying the diary in the diary list semi working.
idoyt committed on May 9

 4b08f7f <>


On May 9, 2022 I created a pop-up used to create new diaries, and displayed the created diaries on the main page, I also styled the login and register pages.

Commits on May 10, 2022

trying to get button to disappear and show a new page with ajax
idoyt committed on May 10

 213f709 <>

.
idoyt committed on May 10

 837bcdc <>

On May 10, 2022 I tried creating a button that would show the diary data and disappear afterwards so that I could dynamically load the data.

Commits on May 18, 2022


working on being able to open diaries
idoyt committed on May 18

 77e7e12 <>

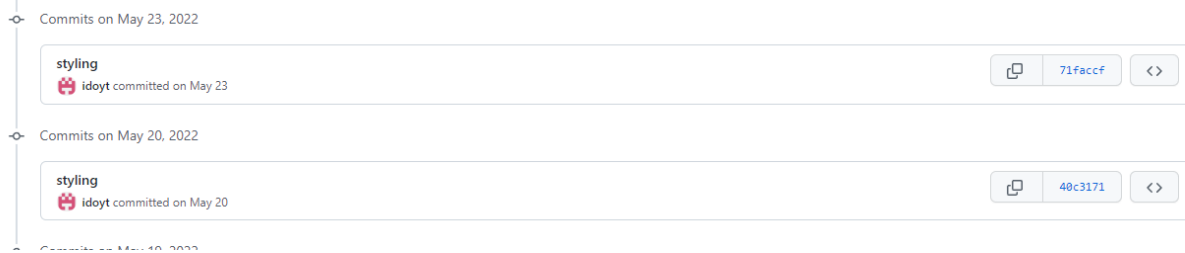
On May 18, 2022 working on the same thing as the May 10, 2022 trying to dynamically load the form.

Commits on May 19, 2022

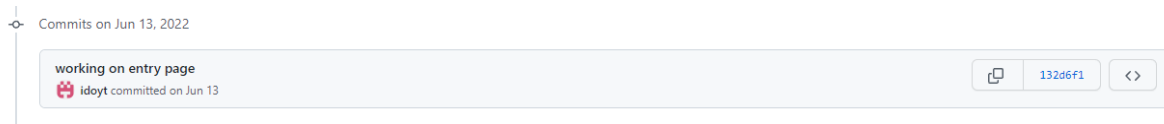
trying to get the button to work
idoyt committed on May 19

 a229e30 <>

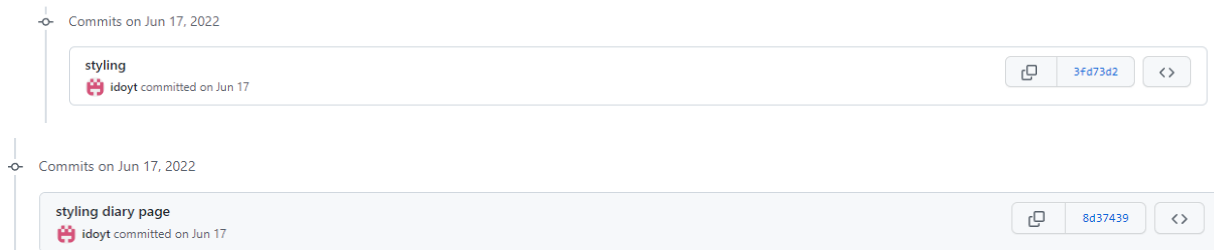
On **May 19, 2022** I scrapped the idea to dynamically load the data and opted to use a button to navigate to the diary page.



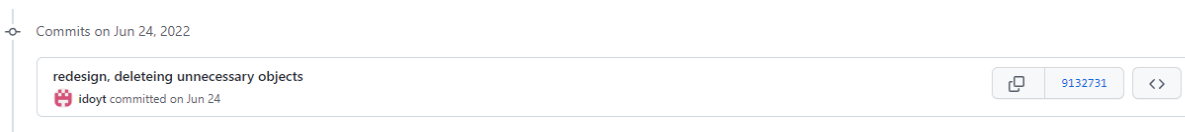
On **May 20, 2022** and **May 23, 2022** I styled my website, to make sure that elements do not overlap each other, and created a handler for when an unauthenticated user tries to access pages that require the user to be logged in.



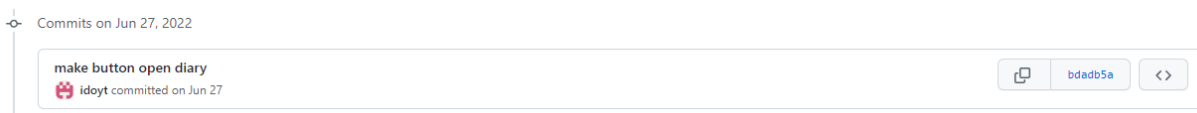
On **Jun 13, 2022**, creating more fields in forms and making them show up on the entering entries page.



On **Jun 17, 2022** styling for diary list and entry list and created a grid to display the diaries.



On **Jun 24, 2022** removed the route where I tried to use js to load the entry data. Removed some of the fields to simplify the website so I can complete it in time.



On **Jun 27, 2022** I tried to query the entries pertaining to a particular diary by querying the entries and the diary individually and listing all the entries where the id of the diary was the same as the id of the diary_id in the entry table.

Commits on Jul 4, 2022

working on creating diaries


 idoyt committed on Jul 4


 42545dd <>

On Jul 4, 2022 working on making the diary creation popup work again.

Commits on Jul 15, 2022

created a create diary page

 idoyt committed on Jul 15

 a3eac77 <>

On Jul 15, 2022 I created a page to create new diaries.

Commits on Jul 16, 2022

testing sqlalchemy relationships


 idoyt committed on Jul 16


 00abbc0 <>

On Jul 16, 2022 I experimented with sqlalchemy relationships to link tables together.

Commits on Jul 18, 2022

entry data display is working.

 idoyt committed on Jul 18

 81c2f71 <>

On Jul 18, 2022 I changed the query so that I could also display the title of the diary so the user would know what diary they were in.

Commits on Jul 29, 2022

diary links go to correct page

 idoyt committed on Jul 29

 6dcff19 <>

On Jul 29, 2022 I turned the diary titles into links and made them take the user to the diary page.

Commits on Aug 2, 2022

made entry template styling entry template

 idoyt committed on Aug 2

 9217017 <>

On Aug 2, 2022 I made a page that lists all the entries and a page that shows the notes and more detailed information of the entry.

Commits on Aug 10, 2022

text editor created working on making it update database


BURNSIDE HIGH SCHOOL committed on Aug 10

 7b9c118 <>

On Aug 10, 2022 I tried to update the notes by querying the entry and using “db.session.update” parsing the text entered in the field which did not work.

Commits on Aug 12, 2022

getting page to display hours instead of just the time values.
BURNSIDE HIGH SCHOOL committed on Aug 12

 5ada8c7 <>

On Aug 12, 2022 I tried using custom filters to parse the entered time values into hours, but it required me to use datetime to use the filter which would make, if I stored the data as datetime it would be repeating the date so I decided to remove activity as it would be too difficult to implement.


Commits on Aug 19, 2022


edit entry updates rows
 idoyt committed on Aug 19

 b085c67 <>

On Aug 19, 2022 I tried "entry.notes = form.notes.data" to edit the entry notes, which still didn't work because of a session error, merging fixed that error. Also making the previously entered data show up in the field would not update this because I called the object too early, moving it to the bottom of the function fixed it.

Commits on Aug 22, 2022


made a delete function for diaries
 idoyt committed on Aug 22

 ef5f831 <>

On Aug 22, 2022 I made a delete function for Diaries. I edited the models so that it would delete all related entries when the diary was deleted.

Commits on Aug 25, 2022

made an edit title page
BURNSIDE HIGH SCHOOL committed on Aug 25

 cad6c0b <>

On Aug 25, 2022 I made a function to edit the diary title.

Commits on Sep 1, 2022


all routes to create/delete/edit/view entries working.
BURNSIDE HIGH SCHOOL committed 28 days ago

 d8e4e57 <>


On Sep 1, 2022 I fixed all my queries to delete/edit/view that were not functional.

Commits on Sep 5, 2022

working on tagging system
18300 committed 23 days ago

 4ba6d4b <>

created a requirements and run.bat
BURNSIDE HIGH SCHOOL committed 24 days ago

 78c71a5 <>

On Sep 5, 2022 I created a requirements and a run.bat which would install all the required packages from the requirements files and run my website. I also started working on a tagging system to increase the complexity of my website and make it easier for the user to tell what's inside the entry before they open it.

Commits on Sep 7, 2022

tags can be added but removes existing tags
BURNSIDE HIGH SCHOOL committed 21 days ago

5d168e7 <>

On Sep 7, 2022 I got the tag system to work, however existing tags would be deleted as all the relationships for the entry would be overwritten. This was because I was getting the row object and creating a new entry every time instead of getting the existing object.

Commits on Sep 9, 2022

adding tags works.
idozt committed 19 days ago

7fa97a2 <>

On Sep 9, 2022 I fixed the issue by getting the existing entry rather than creating a new one.

Commits on Sep 12, 2022

fixed register added create entry button
BURNSIDE HIGH SCHOOL committed 16 days ago

3ee35ac <>

On Sep 12, 2022 my register was broken because I had used the wrong variables in the models page with my check and set password. I also created a route that would create a entry for the current day.

Commits on Sep 13, 2022

remove tag working added 404
BURNSIDE HIGH SCHOOL committed 15 days ago

16ccda1 <>

On Sep 13, 2022 I got the remove tag to work but I later realised that removing one tag from an entry would remove the same tag from another entry. I added a 404 page and handler.

Commits on Sep 14, 2022

styling
idozt committed 14 days ago

a076f0f <>

styling
idozt committed 14 days ago

90384f9 <>

I changed my design from a grid layout to a singular column layout and listed all the diaries like that.

Commits on Sep 25, 2022

error handling
idozt committed 3 days ago

25d9b20 <>








Commits on Sep 24, 2022

register and login errors handling
idozt committed 4 days ago

e2a13d9 <>



From Sep 24, 2022 to Sep 25, 2022 I worked on getting a flask flash to show messages to the user when they enter a field incorrectly and when they've logged in successfully.

Commits on Sep 26, 2022

added favicon idoym committed 2 days ago	 2b81d58 <>
editing home page. idoym committed 2 days ago	 7593a2a <>
adding go back buttons to pages idoym committed 2 days ago	 4ddced1 <>
removing unused template. idoym committed 2 days ago	 b541ff6 <>
added home page, and styling idoym committed 2 days ago	 7af83f7 <>
error handling. idoym committed 2 days ago	 ba4434a <>
HCI, and Pep8. idoym committed 2 days ago	 e093875 <>


On Sep 26, 2022 I worked on setting out my code properly, commenting on what each function did and individual portions of the more complex functions. I remove any unnecessary objects and create error handlers for 401 and 500 errors. I added a homepage detailing what the website is used for and added a favicon.

Commits on Sep 27, 2022

comments and pep8 idoym committed yesterday	 9498ea9 <>
fixing create tag BURNSIDE HIGH SCHOOL committed 2 days ago	 21334f2 <>

On Sep 27, 2022 I fixed a typo which fixed the create tag function. I did some pep8 formatting on my other python files.

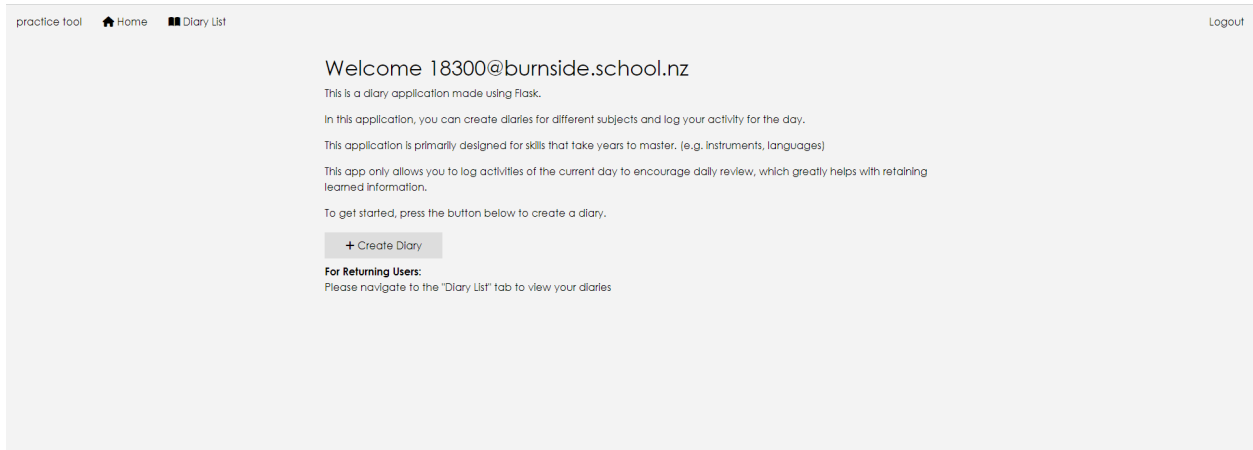
Commits on Sep 28, 2022

ordering entries and diaries in descending and alphabetical respectively idoym committed 1 hour ago	 642a1ae <>
--	--

On Sep 28, 2022 I made sure that the entries and diaries were sorted in descending order and alphabetical order respectively.

Final Iteration of Project

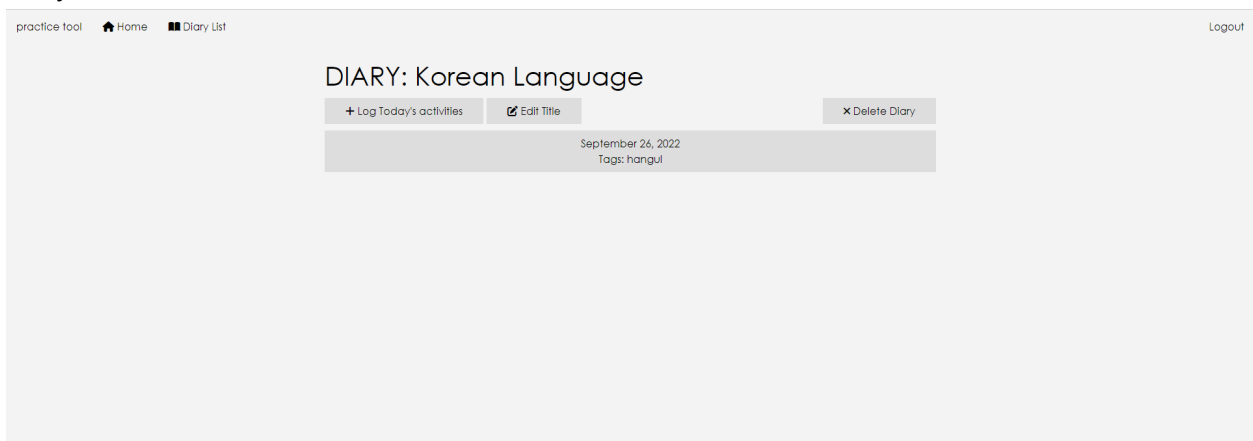
Home



Diary Index



Diary



Entry

practice tool Home Diary List Logout

→ Korean Language → September 26, 2022

September 26, 2022

+ Add Tags

X Delete This Entry

Tags:
hangul X

B I | | | | | | | ?

learning korean writing system
review
- double vowels wo wa ui
- aspirated consonants k t ch p

Save

practice tool Home Diary List Logout

Add a Tag

Tag

Save

Go Back

practice tool Home Diary List Logout

Create Diary

Title

Save

Go Back

Deleting Rows in the database

DELETEING DIARY: "Piano Instrument"

Are you sure you want to delete this diary?
This diary cannot be recovered once deleted.

Delete

Go Back

DELETEING ENTRY FOR "September 26, 2022"

Are you sure you want to delete this entry?
This entry cannot be recovered once deleted.

Delete

Go Back

Authentication

LOGIN

Email

Password

☐ Remember

Don't have an account?

[Click here to Register](#)

Sign In

Register

Email

Password

Confirm Password

Register

Already have an account?

[Click here to Login](#)