INFO1111: Computing 1A Professionalism

2025 Semester 1

Self-Learning Report
Task 2 Advanced: [SQL database powered by Turso]

Student Name: Nathaniel Darren Lim
Student ID: 550391735

Submission number: 2

Vercel website link: https://info-1111-self-learning-advanced-ncg3bdfdv.vercel.app/

Render website link (for PHP): https://strata-pdf-generator.onrender.com

Github link (website): https://github.com/spaceisgreat19/info-1111-self-learning-advanced

Github link (for PHP): https://github.com/spaceisgreat19/strata-pdf-generator

1. Tool/Technology Selection

1.1. What is the tool/technology you have selected?

The tool I selected is a SQL database powered by Turso, which I used to create and manage the database for my project.

Turso is a distributed, edge-based database that improves performance by bringing data closer to users. Since it's built on SQL, it supports familiar concepts like tables and queries, making it easy to use for those with basic database knowledge. It is lightweight and integrates smoothly with modern, serverless applications like those built on Vercel, making it ideal for web apps that need fast and responsive performance. Turso pairs well with Vercel's edge functions, enabling developers to store and access data efficiently at the edge without managing traditional backend infrastructure.

While Turso's edge-first model offers great speed and scalability, it also has some limitations. Syncing data across distributed locations can lead to slight delays, especially with frequent writes. It also lacks some advanced SQL features like stored procedures or triggers, and being relatively new, it has a smaller developer community, which can make troubleshooting a bit harder.

1.2. Why have you selected this tool/technology?

1.2.1. Compare how your chosen tool complements your learning and project in Task 1.

Using a database adds functionality to the website. It allowed me to store important data such as vote counts and maintenance requests, rather than having them reset each time the site was refreshed. Incorporating a database also enabled the creation of a login system, where user credentials and related information could be securely stored and retrieved.

1.2.2. Decide what you will use this new tool to develop or create. Is it an enhancement of your existing project in Task 1? Is it something completely different?

For this project I incorporated my database into three different parts of my system. First, I created a database to store vote counts—users can vote "Yes" or "No," and the totals are saved. Second, I implemented a user authentication system where login and signup details are stored securely in a database. Lastly, I added a maintenance request feature, allowing users to submit requests which are saved and displayed on a dedicated page. Requests can

also be marked as complete or incomplete, with the status updates also saved to the database.

1.3. What benefits do you think learning this tool/technology will give you in the IT industry?

Turso fits well into the web development process by serving as a lightweight, serverless SQL database that works seamlessly with modern frameworks like Next.js. It stores data at the edge, making it ideal for fast, global web apps that need low-latency and real-time interactions, like user logins or voting systems.

Learning how to use Turso builds valuable skills for the IT industry, especially in building scalable, high-performance apps without complex infrastructure. It also strengthens my understanding of data security, SQL, and edge computing—skills that are becoming more important as companies move toward serverless and cloud-first solutions.

STRONG Rating

1. More advanced applications of the tool to your project.

To make my project more advanced, I implemented several key features. I used data from the Votes database to create an interactive bar chart using Recharts, helping users visually understand vote distribution in real time. For security, I added Bcrypt hashing so that user passwords are stored securely as unreadable strings in the database. I also enhanced the maintenance request system by allowing users to sort requests based on whether they've been completed or not. These features improve both functionality and user experience.

2. Alternative tools/technologies - analysis and comparative evaluation

    1. Identify 2 alternative tools/technologies that can be used instead of the one you chose for your topic.

Two alternative tools that can be used instead of Turso are Supabase and PlanetScale. Supabase is an open-source backend-as-a-service built on PostgreSQL. It provides built-in features like user authentication, file storage, and real-time updates, which makes it a great all-in-one solution for building and launching full-stack apps quickly. PlanetScale is a serverless MySQL-compatible database designed for high scalability. Its advanced capabilities, like branching and non-blocking schema changes, make it ideal for large, growing applications that require stability and frequent database changes without downtime.

2. Comparative Analysis

2.1. Describe situations in which both your tool/technology and each of the identified alternatives would be preferred over the others.

Turso is best suited for smaller, fast-loading apps where edge performance matters, like real-time dashboards or serverless tools, because it stores data close to the user. It's simple to set up and requires little maintenance, which makes it ideal for personal projects or lightweight applications.

Supabase is a better fit when you need more than just a database. It includes features like user auth, file storage, and live updates, so it's ideal for MVPs or early-stage apps where you want a full backend with minimal setup. Its integration with PostgreSQL also gives developers access to a familiar and powerful relational database.

PlanetScale shines in large, production-level environments. Its MySQL compatibility and advanced features like branching and online schema changes make it perfect for apps that demand stability and scalability. It's especially useful when working with multiple developers or managing frequent schema updates without affecting users.

2. Journal of self-learning activities and reflections.

22/04/25
I began by trying to integrate a database into my voting system to store the number of vote. I started by trying to use Prisma, but after several unsuccessful attempts, I couldn't get it to work. I did some research and eventually decided to use Turso.

First, I set up my database with Turso, and then I created the Post and Get backend routes under src/api. Then I updated my Vote homepage to reflect the new changes. There were multiple bugs and it took a lot of trial and error. But with persistence I was able to get everything working in the end.

24/04/25
After successfully setting up my first database, I decided to build a login system for my finance page using cookies. To start, I spent some time studying how cookies work and how to implement them in my project.

Next, I built the backend for the login system, creating a new database where users could securely store their account details. I also developed both a login page and a sign-up

page, allowing users to create accounts and access their information. The process was quite similar to building the vote database, so I was able to apply many of the same concepts.

25/04/25

I spent more time studying how cookies work, especially in relation to session management. After a user logs in, a session token is generated, which is necessary for accessing the finance page. To manage this, I created middleware to handle the session token.

Initially, the finance page didn't work as expected, even after I added the cookies. However, after refining and updating my middleware, I was able to resolve the issue, and the page started functioning correctly.

27/04/25

Next, I began researching how to integrate PHP into my project. I wanted to use PHP to generate a PDF levy notice. I explored several options, including Vercel PHP and InfinityFree. I initially tried InfinityFree, but ran into issues uploading my file. Eventually, I settled on using Render. I created a file called generate-pdf.php, and uploaded it to the platform. After confirming that PDF generation was functioning correctly, I proceeded to build the frontend for the levy page. The development process was straightforward since I was already familiar with creating frontend pages.

28/04/25

I created a new database and integrated it with the maintenance feature. Since I had already gone through the process before, setting it up was relatively straightforward. I also took the opportunity to improve the formatting of the maintenance page for a cleaner and more user-friendly layout.

30/04/25

After confirming everything worked locally, I pushed my code to Git and deployed it on Vercel. I ran into several bugs during deployment, mainly caused by leftover files from Prisma and my Vote backend setup. After some troubleshooting, I managed to deploy the project successfully. I then focused on refining the interface—formatting the pages with Tailwind CSS and using Recharts to create an interactive bar graph on the votes page. I attempted to add a database to the finance section, but the code became difficult to manage when combined with cookie-based login logic. I also tried implementing a feature that would allow users to create new issues to vote on, but I ran into problems connecting it to the database and wasn't able to get it working in time.

31/04/25

Once I completed the project, I focused on writing the report, where I documented my results and explained the development process in detail.

3. Self-learning Plan

3.1. **Goals**:

Your goals for this report are to:

1. Demonstrate your knowledge (understanding) of the tool/technology you have selected by carrying out the steps specified in the instructions.
2. Demonstrate your skill in applying that knowledge to create artefacts using your tool/technology by carrying out the steps specified in the instructions.
3. Analyse, compare and evaluate your tool/technology in a broader context by carrying out the steps specified in the instructions.
4. Evaluate your self-learning processes.

3.2. **Resources**

3.2.1. **Information resources required**

TITLE	AUTHOR	PURPOSE	REFERENCE & LINK
What is the title of the resource?	*Who is the author of the resource?*	*What will you use this resource for (e.g. learning how to create a project).*	*Include the reference for the source and the link (shorten the link if it is long).*
First impressions with Turso database	Elton Minetto	Using Turso databse	*https://dev.to/eminetto/first-impressions-with-the-turso-database-5g0*
How to develop serverless PHP application with PostgreSQL database with Vercel and neon.tech for free	Daniil Bazhenov	Integrating PHP using Neontech	https://dev.to/dbazhenov/how-to-develop-serverless-php-application-with-postgresql-database-with-vercel-and-neontech-for-free-dgl

Understanding cookies	Lydia Hallie	Using cookies	https://vercel.com/blog/understanding-cookies
Thoughts on render.com	Siegfriedson	Integrating PHP by using Render	https://medium.com/sort-of-like-a-tech-diary/thoughts-on-render-com-214176365ecc
Turso	Unknown	Set up integration on Turso	https://developers.cloudflare.com/workers/databases/native-integrations/turso/?utm_source=chatgpt.com
Cookies example	Vercel	Adding cookies	https://vercel.com/templates/next.js/cookies
How to deploy PHP APP on Vercel	Jarvis Silva	Integrating PHP	https://hostwizly.com/deploy-php-app-on-vercel?utm_source=chatgpt.com

3.3. **Schedule**

STEPS NO	DESCRIPTION	DATE
1	Add my vote database	22/04/25
2	Create a login database and page	24/04/25
3	Create cookies and update finance page	25/04/25
4	Create PHP back end and levy page	27/04/25
5	Update maintenance with database	28/04/25
6	Upload project to Vercel	30/04/25
7	Complete report	31/04/25

4. Results: evidence of the steps you have attempted from the instructions

Vercel provides several data storage options, including Vercel Postgres for structured data, Vercel KV for fast key-value access, Edge Config for globally distributed settings, and Vercel Blob for handling file uploads. Unlike traditional servers that store data locally, Vercel uses stateless serverless functions, which require external or managed storage solutions. This makes it easier to scale apps, but also means developers have less control over the servers.

For my database, I first created and structured the database with tables such as users (containing username and password_hash columns) and maintenance_requests (storing fields like apartment, issue, and completed).

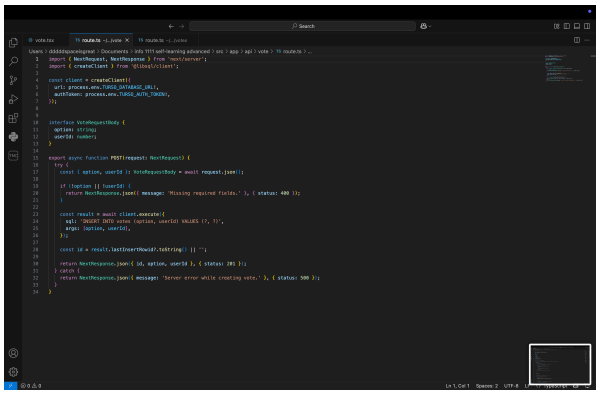
When a user performs an action—such as signing up, voting, or submitting a maintenance request—the frontend sends a POST request to an API route (/api/signup, /api/vote, /api/log-maintenance). These routes are handled by Vercel serverless functions, which use the libsql client to execute INSERT statements and store the data in Turso.

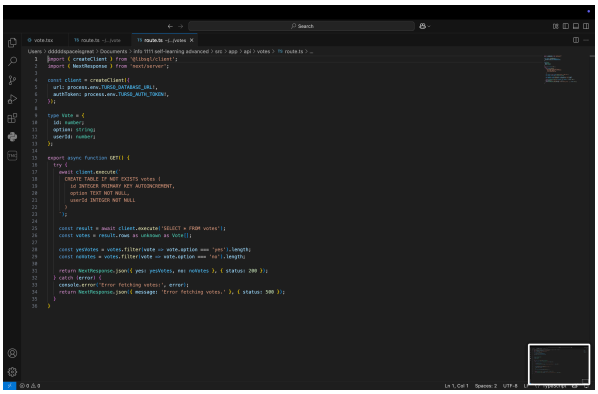
To retrieve and render stored data (e.g., displaying votes or maintenance logs), I use GET requests to corresponding API endpoints. These API routes execute SELECT queries against Turso, and return the results as JSON to the frontend.

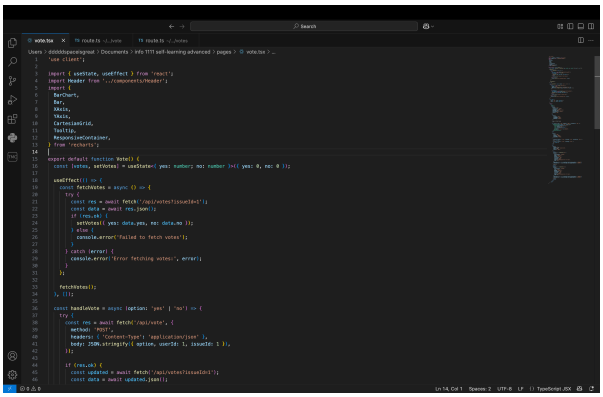
1. First, I created an account on Turso by running the command “turso auth login”. After logging in, I created a new database using “turso db create”. I then created a table called votes to store the voting data.

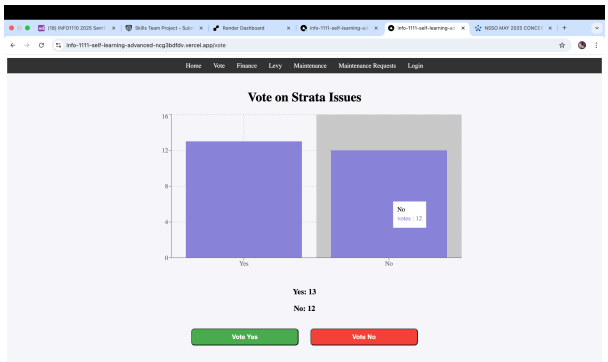
```
→ PRAGMA table_info(votes);
CID     NAME        TYPE        NOTNULL     DFLT VALUE     PK
0       id          INTEGER     0           NULL           1
1       option      TEXT        1           NULL           0
2       userId      INTEGER     1           NULL           0
3       issueId     INTEGER     1           1              0
→ SELECT * FROM votes;
ID      OPTION      USERID      ISSUEID
1       yes         1           1
2       yes         1           1
3       no          1           1
4       no          1           1
5       yes         1           1
6       yes         1           1
7       yes         1           1
8       yes         1           1
9       no          1           1
10      no          1           1
11      yes         1           1
12      no          1           1
13      yes         1           2
14      no          1           2
15      yes         1           1
16      no          1           3
17      no          1           1
18      no          1           1
19      yes         1           1
20      no          1           1
21      no          1           1
22      yes         1           1
23      yes         1           1
24      yes         1           1
25      no          1           1
```

2. After creating the votes database, I created the api/vote route to handle POST requests for recording user votes and the api/votes route to handle GET requests for retrieving the current vote counts. I then updated the frontend to display the voting functionality, and integrated basic CSS to improve the page's appearance. Additionally, I included an interactive bar-chart using Recharts, which shows the number of votes for each option.









3. I created a login and sign-up system. First, I set up a database called "users" to store all login credentials securely. Then, I built two web pages: the sign-up page allows users to register by creating a username and password, while the login page lets them log in with those credentials and generates a session token in the form of a cookie.

In the backend, the API route at api/signup checks if the user already exists and creates the user only if the username is unique. Passwords are hashed using bcrypt before being stored, so they appear as random strings in the database. The api/login route validates the credentials and, if correct, generates a session token and sets it as a cookie for authentication.

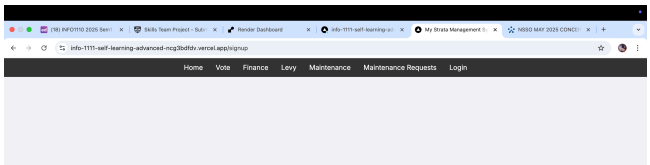
```
→  PRAGMA table_info(users);
CID     NAME        TYPE        NOTNULL     DFLT VALUE     PK
0       id          INTEGER     0           NULL           1
1       username    TEXT        1           NULL           0
2       password    TEXT        1           NULL           0
→  SELECT * FROM users;
ID     USERNAME     PASSWORD
1      test         $2b$10$ymGZs0P/utGcAPannW7BjeWI4og.GQQyoW7p7gpa10M8tA.uAHTi6
```

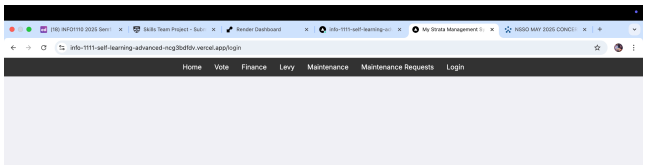
```typescript
import { NextRequest, NextResponse } from 'next/server';
import { createClient } from '@libsql/client';
import bcrypt from 'bcryptjs';

const client = createClient({
  url: process.env.TURSO_DATABASE_URL,
  authToken: process.env.TURSO_AUTH_TOKEN,
});

export async function POST(request: NextRequest) {
  try {
    const { username, password } = await request.json();

    if !username || !password) {
      return NextResponse.json({ message: 'Missing username or password.' }, { status: 400 });
    }

    await client.execute(`
      CREATE TABLE IF NOT EXISTS users (
        id INTEGER PRIMARY KEY AUTOINCREMENT,
        username TEXT UNIQUE NOT NULL,
        password TEXT NOT NULL
      )
    `);


    const existingUser = await client.execute({
      sql: 'SELECT * FROM users WHERE username = ?',
      args: [username],
    });

    if (existingUser.rows.length > 0) {
      return NextResponse.json({ message: 'Username already taken.' }, { status: 400 });
    }


    const hashedPassword = await bcrypt.hash(password, 10);


    await client.execute({
      sql: 'INSERT INTO users (username, password) VALUES (?, ?)',
      args: [username, hashedPassword],
    });

    return NextResponse.json({ message: 'User created successfully!' }, { status: 201 });
  } catch (error) {
```

```typescript
import { NextRequest, NextResponse } from 'next/server';
import { createClient } from '@libsql/client';
import bcrypt from 'bcryptjs';
import crypto from 'crypto';

const client = createClient({
  url: process.env.TURSO_DATABASE_URL,
  authToken: process.env.TURSO_AUTH_TOKEN,
});

export async function POST(request: NextRequest) {
  try {
    const { username, password } = await request.json();

    if !username || !password) {
      return NextResponse.json({ message: 'Missing username or password.' }, { status: 400 });
    }

    const result = await client.execute({
      sql: 'SELECT * FROM users WHERE username = ?',
      args: [username],
    });

    if !result.rows.length === 0) {
      return NextResponse.json({ message: 'User not found.' }, { status: 404 });
    }

    const user = result.rows[0];

    if !!user.password) {
      return NextResponse.json({ message: 'Password not found.' }, { status: 400 });
    }

    const isPasswordValid = await bcrypt.compare(password, String(user.password));

    if !!isPasswordValid) {
      return NextResponse.json({ message: 'Invalid credentials.' }, { status: 401 });
    }

    const sessionToken = crypto.randomBytes(64).toString('hex');

    const headers = new Headers();
    headers.append(
      'Set-Cookie',
      serialize('session_token', sessionToken, {
```

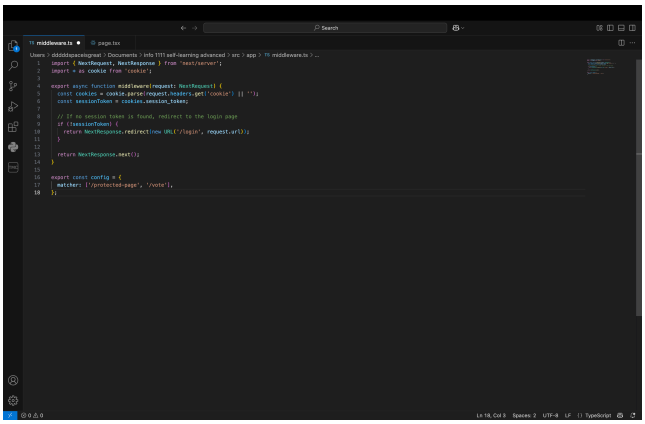
```typescript
'use client';

import { useState } from 'react';
import Header from '../../components/Header';
import Link from 'next/link';
import '../../styles/globals.css';

export default function Login() {
  const [username, setUsername] = useState('');
  const [password, setPassword] = useState('');
  const [error, setError] = useState('');

  const handleSubmit = async (e: React.FormEvent) => {
    e.preventDefault();

    const response = await fetch('/api/login', {
      method: 'POST',
      headers: {
        'Content-Type': 'application/json',
      },
      body: JSON.stringify({ username, password }),
    });

    const data = await response.json();

    if !response.ok) {
      window.location.href = '/';
    } else {
      setError(data.message);
    }
  };

  return (
    <div className="min-h-screen bg-gray-100 flex flex-col">
      <Header />
      <div class Name="flex-grow flex items-center justify-center">
        <div class Name="bg-white p-8 rounded-2xl shadow-md w-full max-w-md text-black">
          <h2 class Name="text-2xl font-bold mb-6 text-center">Login</h2>
          <form onSubmit={handleSubmit} class Name="space-y-4">
            <div>
              <label class Name="block mb-1">Username</label>
                <input
                  type="text"
                  placeholder="Try 'test'"
                  value={username}
                  onChange={(e) => setUsername(e.target.value)}
```

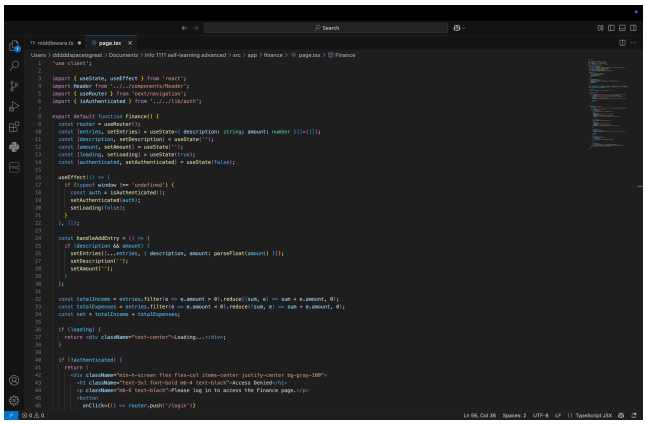
```typescript
'use client';

import { useState } from 'react';
import Header from '../../../components/Header';

export default function Signup() {
  const [username, setUsername] = useState('');
  const [password, setPassword] = useState('');
  const [error, setError] = useState('');
  const [success, setSuccess] = useState('');

const handleSubmit = async (e: React.FormEvent) => {
  e.preventDefault();

  const response = await fetch('/api/signup', {
    method: 'POST',
    headers: {
      'Content-Type': 'application/json',
    },
    body: JSON.stringify({ username, password }),
  });

  const data = await response.json();

  if (response.ok) {
    setSuccess('Signup successful! You can now login.');
    setError('');
  } else {
    setError(data.message);
    setSuccess('');
  }
};

return (
  <div>
    <Header />
    <div class Name="min-h-screen flex items-center justify-center bg-gray-100">
      <div class Name="bg-white p-8 rounded-2xl shadow-md w-full max-w-md text-black">
        <h2 class Name="text-2xl font-bold mb-6 text-center">Sign Up</h2>
        <form onSubmit={handleSubmit}>
          <div class Name="mb-4">
            <label class Name="block text-gray-700">Username</label>
              <input
                type="text"
                value={username}
                onChange={(e) => setUsername(e.target.value)}
```

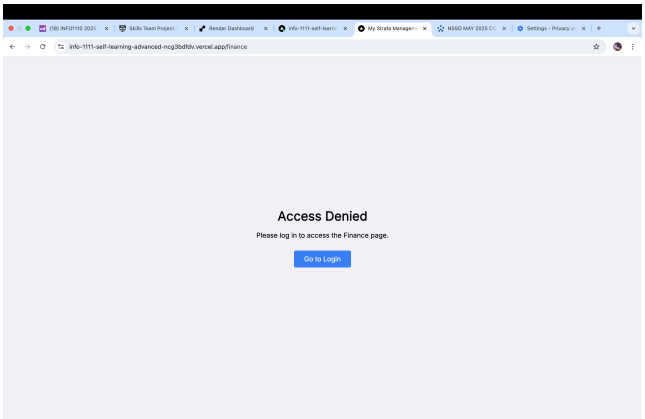


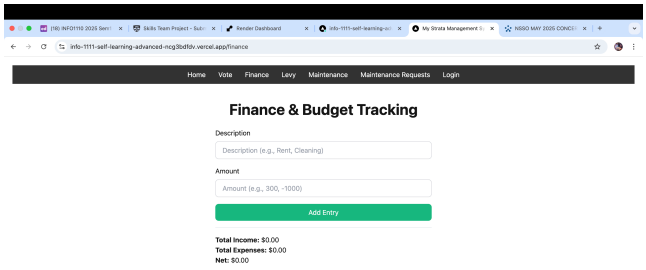


4. After setting up the login system, I updated the Finance page to require user authentication before access. I used middleware.ts to check if a user is logged in by verifying the presence of a session token stored in a cookie. If the token is not found, the page will ask users to login. I also applied Tailwind CSS to enhance the visual appearance and improve the usability of the Finance page.

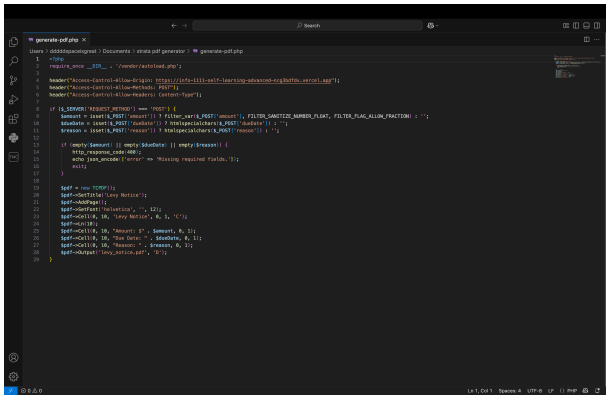


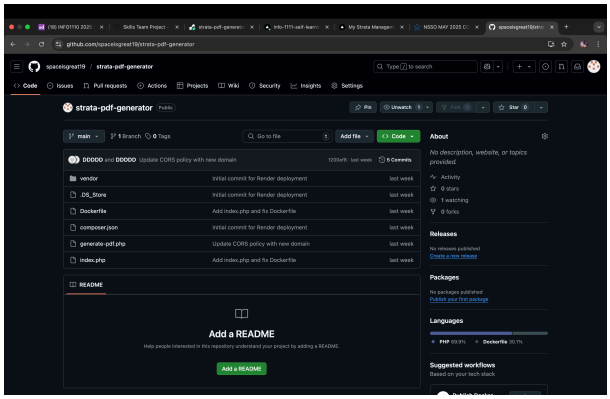


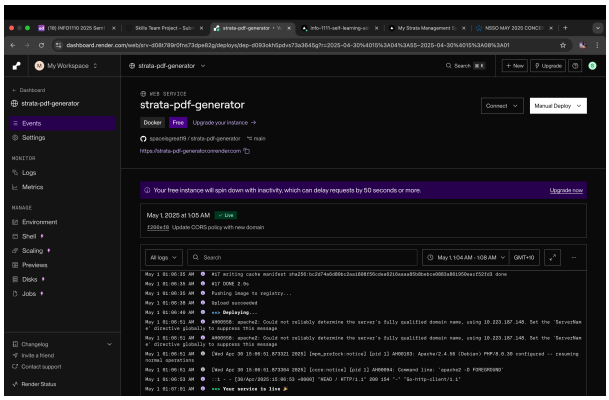


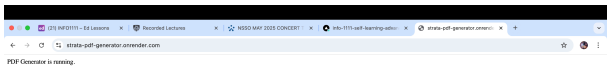


5. To incorporate PHP into my website, I created a file called generate-pdf.php to generate a PDF levy using the TCPDF library. Afterwards, I pushed the project to a separate Git, including the vendor directory. Finally, I deployed the project on Render to make the PDF generation functionality live. (The website may take a while to load)

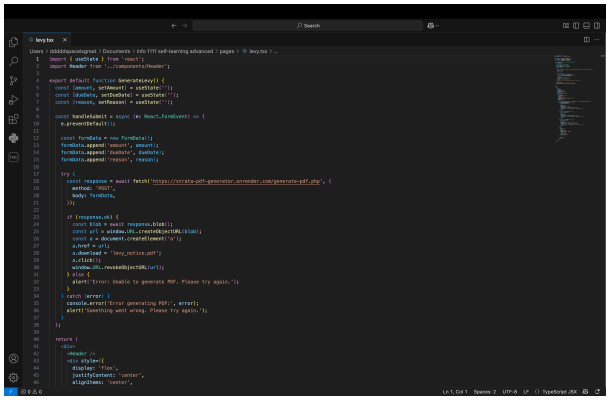


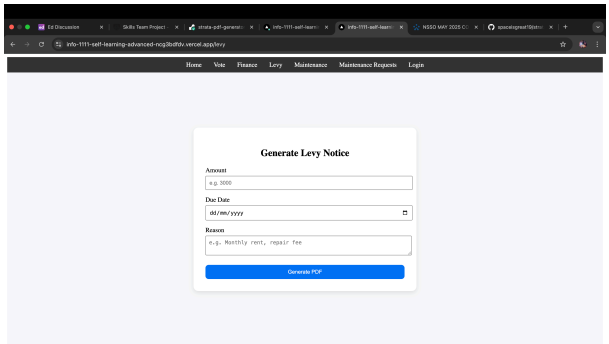


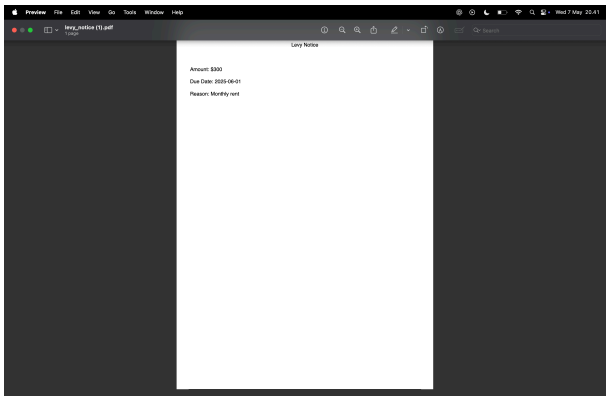




6. Since the backend for generating PDFs was completed, I proceeded to create a levy page. This page allows users to input data, and based on that data, a PDF is generated. (The download may take a while to load)

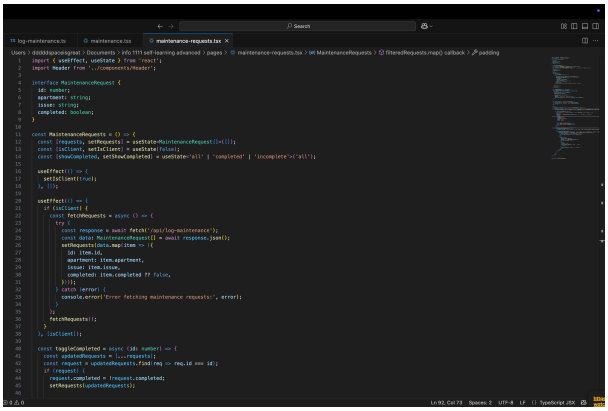


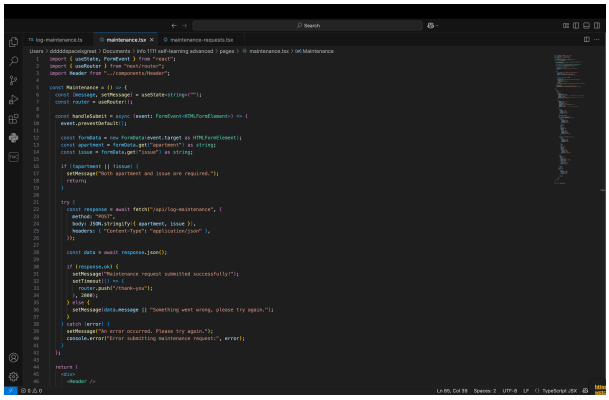


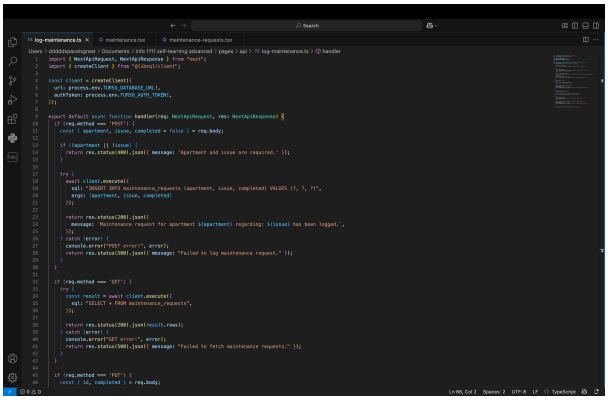


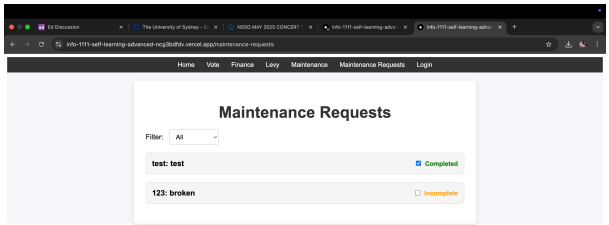
7. I created a new Turso database called maintenance-requests to store maintenance request data. I implemented a api/log-maintenance endpoint to handle both POST and GET requests, allowing users to submit and retrieve maintenance issues. The submission form includes an option for users to mark whether a request is completed. I also updated the frontend to reflect these changes and improved the page layout.

```
→  PRAGMA table_info(maintenance_requests);
CID     NAME          TYPE        NOTNULL     DFLT VALUE     PK
0       id            INTEGER     0           NULL           1
1       apartment     TEXT        0           NULL           0
2       issue         TEXT        0           NULL           0
3       completed     BOOLEAN     0           FALSE          0
→  SELECT * FROM maintenance_requests;
ID     APARTMENT     ISSUE      COMPLETED
1      test          test       1
2      123           broken     0
```









5. Evaluation

5.1. Knowledge and skills

QUESTIONS	YOUR ANSWER
1. To what extent did you reach the goals for this report?	I was able to create a database, use cookies, and intergrate PHP
2. What barriers did you face in reaching the goals?	Knowledge gap, bugs in code
3. What worked well for you in doing the report?	Online resources, AI
4. What was frustrating?	I was over-ambitious, I spent a lot of time adding a lot of functions, but they didn't end up working.
5. Other?.....	

1. Reflect on your use of generative AI. Keep in mind, for this assignment any use of AI to write code and help with your learning is allowed an encouraged, however the University does not condone the use of the DeepSeek AI model. All students have access to Microsoft Copilot for free, read more here: How to use generative artificial intelligence in the classroom - The University of Sydney. You may also choose to use an AI-powered code editor (IDE) like Cursor or Windsurf.

QUESTIONS	YOUR ANSWER
1. To what extent did you use generative AI in this task? Provide an example of where you used AI.	I used AI to help me find good resources, explain difficult concepts, troubleshoot coding issues, help code, and refine my report.
2. What kinds of prompts were helpful? Did you ask follow-up questions? For what kinds of things?	Help me explain this concept in simple terms, Help me fix my code, Please refine my text. I asked follow up questions when my problems aren't fully solved yet.
3. What kinds of prompts were not helpful? Were there any instances of the AI model giving output which did not work or was not expected?	Sometimes the code doesn't work or doesn't make sense.
4. Imagine that AI was unavailable or its use was prohibited. Do you think you would you have been able to achieve the goals for this task? Why / why not?	I could have done it without AI, but it would have been much more difficult. I would have had to spend significantly more -me solving problems and understanding complex concepts on my own.
5. Do you think that this task has improved your confidence with using AI? What new use cases of generative AI did you discover, and what applications do you think they will have for you in the future?	Yes, it taught me that AI can be a useful tool, but it has certain limitations. While AI can provide guidance, there are times when it's better to figure things out ourselves. In the future, I can use AI to support my work and studies, but I will also be mindful of when to rely on my own problem-solving skills.

6. Identify and compare the aspects of the task that AI could help with, and the aspects that AI could not help with. Do you think that in the future, AI capabilities might expand even further to cover more of those things? How would you expect your role to adapt, if you imagine that this task was part of your job?	AI helped me with difficult concepts and troubleshooting my code, but it sometimes gave unclear or incorrect advice. In the future, AI capabilities can definitely expand. I would diversify myself to learn new things so I can enhance my skills.
7. Other?.....	

3. Self-learning learning processes

QUESTIONS	YOUR ANSWER
1. What worked?	Online resources, AI
2. What didn't?	Doing things without a plan, making things too complicated
3. What would you do differently?	Create a proper plan before starting
4. What did you learn about yourself?	I can be over ambitious sometimes
5. What recommendation would you make to your future self?	Its okay to dream big, but make sure to start small
6. What would you recommend to someone else?	Use AI to help troubleshoot code
7. Other?.....	

6. Learning sources

Learning Source - What source did you use? (Note: Include source details such as links to websites, videos etc.). Contribution to Learning - How did the source contribute to your learning (i.e. what did you use the source for?). You may use information from your Journal for this.

Learning Source - What source did you use? (Note: Include source details such as links to websites, videos etc.).	Contribution to Learning - How did the source contribute to your learning (i.e. what did you use the source for)?

https://dev.to/eminetto/first-impressions-with-the-turso-database-5g0	Using Turso database
https://developers.cloudflare.com/workers/databases/native-integrations/turso/?utm_source=chatgpt.com	Set up integration on Turso
https://vercel.com/blog/understanding-cookies	Adding cookies
https://vercel.com/templates/next.js/	Example of cookies
https://dev.to/dbazhenov/how-to-develop-serverless-php-application-with-postgresql-database-with-vercel-and-neontech-for-free-dgl	Integrating PHP using Neontech
https://medium.com/sort-of-like-a-tech-diary/thoughts-on-render-com-214176365ecc	Integrating PHP on Render
https://hostwizly.com/deploy-php-app-on-vercel?utm_source=chatgpt.com	Integrating PHP on Vercel

Bibliography

[1] The University of Sydney, _Referencing and citation styles: IEEE,_ 2022, see https://libguides.library.usyd.edu.au/c.php?g=508212.