

National College of Ireland

Computing

Cyber Security

2023/2024

Lee Campbell

X20115075

X20115075@student.ncirl.ie

PennyWise

Technical Report

Contents

Executive Summary	1
1.0 Introduction	2
1.1. Background	2
1.2. Aims	2
1.3. Technology	2
The technology stack:	3
Development and CI/CD	4
1.4. Structure	4
2.0 System	4
2.1. Requirements	4
2.1.1 Use Case Diagram	4
2.1.1.1 Requirement 1: User Registration (Passwordless Sign-up with OTP)	4
2.1.2 Use Case Diagram	6
2.1.2.1 Requirement 2: User Login (Passwordless Sign-in with OTP)	6
2.2. Design & Architecture	7
2.3. Graphical User Interface (GUI)	16
2.4. Testing	24
2.5. Evaluation	25
3.0. Conclusions	25
3.0 Further Development or Research	26
4.0 References	27
5.0 Appendices	27
5.1. Project Proposal	27
5.2. Reflective Journals	38

Executive Summary

This report provides a technical overview of the PennyWise application, which aims to integrate personal banking and AI for financial analytics and insights. The purpose of the report is to examine and document the background, objectives, problem areas, development, system design, implementation, testing and evaluation of this application.

This report has highlighted the benefits of the current architecture, using Server Side Rendering, the benefits of the technology stack used, the use cases, current implementation and security considerations going forward and the current infrastructure and code.

This report provides a technical review of the PennyWise application, which aims to merge personal banking with AI capabilities for financial analysis and insights. The report will explore and detail the project's background, goals, challenges, development process, system design, implementation, testing, and evaluation.

Key highlights of the report include:

- **Background and Objectives:** The report delves into the how and why of the PennyWise application and how the current objectives came to be.
- **Architecture and Technology Stack:** Emphasizes the advantages of the current architecture, particularly the use of Server Side Rendering, and discusses the benefits of the chosen technology stack.
- **Use Cases and Implementation:** The report outlines various use cases, the current implementation status, and how the application integrates AI for financial analytics.
- **Security Considerations:** Focus on security considerations in the application's design and implementation, in an aim to highlight the importance of safeguarding user financial data.
- **Infrastructure and Code:** It provides insights into the current infrastructure and codebase, offering an overview of the technical aspects of the project.

1.0 Introduction

1.1. Background

Traditional financial management tools lack dynamic insights – PennyWise wants to allow people to make more informed decisions on their money, and how to manage it.

PennyWise aims to give customers intelligent advice and personalised money management techniques, similar to how diagnostics for cars can identify problems. This initiative takes on a new relevance in the current context of the rising cost of living issue, which is particularly evident in Ireland. The overall objective is to develop an application that not only educates but also empowers people to manage their finances, in an aim to transform personal banking.

1.2. Aims

The primary aim of the PennyWise project is to develop a financial management application that provides users with personalised and dynamic insights on how best to manage their finances effectively.

While PennyWise strictly does not give financial advice, it aims to provide insights – which may help an end user in learning how to be more financially minded.

1.3. Technology

Technology used with this project, is **NextJS**, a full-stack offering, a react-based framework built on JavaScript for building web applications. This framework uses **React**

and **TypeScript** for the Frontend, while NextJS offers functionality that makes this application full-stack. **TypeScript**, which is a superset of JavaScript, brings static typing and modern language features.

NextJS also provides out of the box functionality for server-side rendering, static site generation, and creating API routes, enabling my application, PennyWise to implement features as needed, both on the server and the frontend.

The technology stack:

Frontend

- **Vercel**: A platform for deploying frontend applications, commonly NextJS Applications,
 - Automatic Deployments
 - Multiple Environments
 - Edge Functions
 - Instant Scaling and Global CDN Distribution
 - Can also host databases and other backend applications.
- **TailwindCSS**: A utility-first CSS framework used for designing the user interface, follows a CSS-in-JS approach.

User Authentication

- **Kinde**: A user authentication provider used to manage user sign-ups, logins, and secure access to the application.

Database

- **PostgreSQL**: An ACID-compliant, open-source relational database management system used for storing and managing application data.
- Despite initial considerations, **Prisma** was not used due to issues with the generated types, which led to its removal from the project.
- **This also led to the retiring of the PostgreSQL database.**

API and Data Management

- **Zod**: A TypeScript-first schema validation library, used for defining and validating data schemas.

Additional Libraries and Tools

- **@radix-ui/react-avatar** and **@radix-ui/react-scroll-area**: Components from Radix UI used for building accessible UI components – brought in from shadcn component library.
- **lucide-react**: A library for SVG icons.
- **recharts**: A composable charting library for React.
- **react-markdown** and **rehype-highlight**: Used for rendering and syntax highlighting markdown content.
- **sanitize-html**: Used for sanitizing HTML to prevent XSS attacks.

Development and CI/CD

- **GitHub Actions:** Used for continuous integration and deployment (CI/CD), automated workflows for building, testing, and deploying the application alongside Vercel.
- **Prettier:** A code formatter to maintain consistent code style.
- **ESLint:** A tool for identifying and fixing issues in JavaScript code.
- **Playwright:** A testing library used for end-to-end testing, ensuring the application's functionalities work as expected across different browsers.

1.4. Structure

This report will follow a structured format, beginning with an section on system requirements, followed by design and architecture, implementation, testing, evaluation, conclusions, further development or research, references, and appendices.

Each section will provide insights into the different aspects of the project, including task generation, milestone planning, system design, and testing strategies. Diagrams will be shown in relevant sections where appropriate for added context.

2.0 System

2.1. Requirements

- **User Registration:** Users sign up for the PennyWise application using email and OTP for authentication.
- **User Login:** Existing users can log in to the PennyWise application using email and OTP.
- **Bank Connectivity:** Integration with multiple banks in Ireland.
- **Transaction Data:** Ability to fetch transaction data securely.
- **Dashboard:** Authorised users can view a dashboard displaying financial analytics and insights.
- **AI Chatbot:** Implement an AI-driven chatbot (PennyWise) to provide financial insights about fetched transaction data.
- **Secure Data Handling:** Encryption of sensitive user data both in transit and at rest.
- **Error Handling:** No exposure of backend processes, graceful handling of errors.

2.1.1 Use Case Diagram

2.1.1.1 Requirement 1: User Registration (Passwordless Sign-up with OTP)

Description & Priority

This requirement entails users registering for the PennyWise application without passwords, using a one-time password (OTP) for authentication.

Priority: High

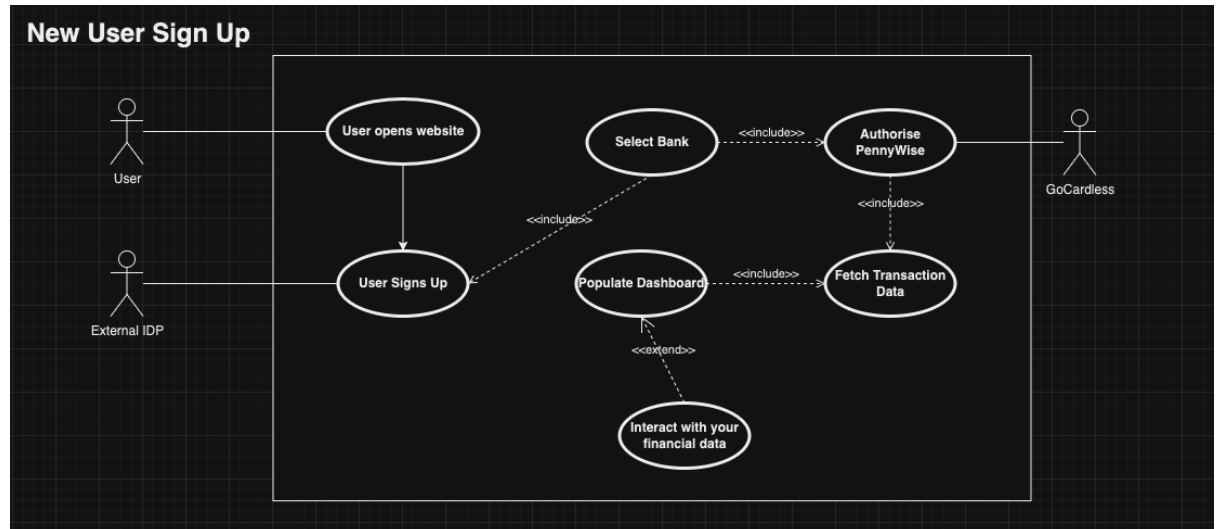
Use Case Scope:

The scope of this use case is to allow users to register for the PennyWise application passwordlessly using a One Time Password (OTP)

Description:

This use case involves users entering their email address, first and last name, receiving an OTP, verifying their email through OTP, selecting a bank, and gaining access to the application's dashboard.

Use Case Diagram:



Flow Description:

Precondition:

The user is accessing the PennyWise registration page.

Activation:

This use case starts when a new user navigates to the registration page.

Main flow:

The user enters their first and last name, email address. The system generates a one-time password (OTP) and sends it to the user's email. The user enters the OTP received in their email to verify their email address. The user selects their bank from the available options. The user successfully registers and gains access to the dashboard.

Alternate flow:

Register with Google

Incorrect email address format: The system prompts the user to enter a valid email address.

OTP not received: The system allows the user to request a new OTP or provides alternative verification methods. Exceptional flow:

Email delivery failure: The system informs the user that there was an issue sending the OTP and advises them to check their email address or try again later.

Termination

The dashboard is shown to user as user is logged in and credentialed, until token expiry is activated.

2.1.2 Use Case Diagram

2.1.2.1 Requirement 2: User Login (Passwordless Sign-in with OTP)

Description & Priority

This requirement involves existing users logging into the PennyWise application without passwords, using a one-time password (OTP) for authentication.

Priority: High

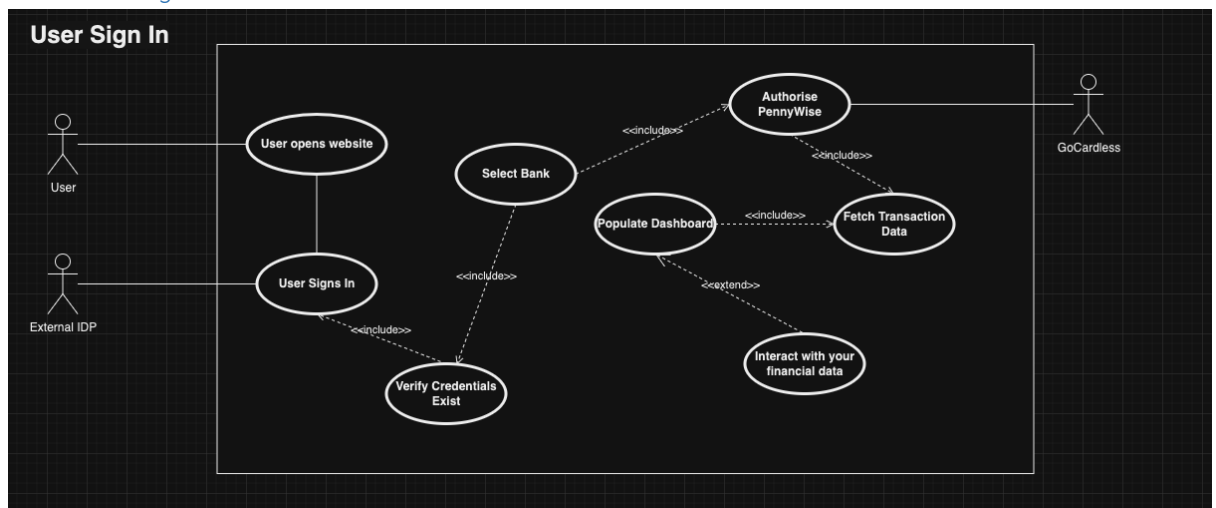
Use Case Scope:

This use case allows existing users to log into the PennyWise application without using passwords, using a One Time Password(OTP) for authentication.

Description:

This use case involves existing users entering their email address, receiving an OTP, verifying their identity through OTP, selecting a bank, and accessing the application's dashboard.

Use Case Diagram:



Flow Description:

The user enters their email address or click continue with Google. The system generates a one-time password (OTP) and sends it to the user's email. The user enters the OTP received in their email to verify their identity. The user selects their bank from the available options. The user gains access to the dashboard

Precondition:

The user is accessing the PennyWise login page.

Activation:

This use case starts when an existing user navigates to the login page. Main flow:

Alternate flow:

Login with Google

Incorrect email address format: The system prompts the user to enter a valid email address.

OTP not received: The system allows the user to request a new OTP or provides alternative verification methods. Exceptional flow:

Email delivery failure: The system informs the user that there was an issue sending the OTP and advises them to check their email address or try again later.

2.1.3 Data Requirements:

- User must be logged in.
- User's GoCardless auth token must be saved in the user session.

2.1.4 User Requirements:

- User should have an email addressed
- User should have a supported financial institution with transaction history.

2.1.5 Environmental Requirements:

- TBD

2.1.6 Usability Requirements:

- Application should respond quickly.
- Ensure non-exposure of sensitive data.
- Implemented security measures.
- Design the application interface to look aesthetically pleasing.

2.2. Design & Architecture

Sample Diagram:



Current Design: secure and scalable server-side rendered (SSR) architecture:

- Authentication layer is managed by a Kinde server side.
- API layer serves as a shield, encapsulating and abstracting external API interactions
- authentication process involves storing of access tokens within user sessions objects, created and contained within a database.
- The server manages these user sessions, associating them with the corresponding client requests

- If user is authenticated(client can make requests) – users gain access to application features, including conversations with PennyWise(ChatGPT) and retrieval of financial data.

Next Steps:

- Comprehensive Error Handling
- Secure by Design – adding auth to all routes
- Caching
- Data encryption and bolstered security.

Chat UI Functionality:

- Use Chat hook provided by Vercel to communicate with ChatGPT AI (OpenAI).
- Setting the ID as the date, the context tells the Agent what to be (sample).
- the useChat hook to manage chat-related functionality such as handling user input, sending messages, and displaying messages.
- Handle submit supplies the information to my defined API route and sends it back to OpenAI – which then streams a response.

```
export default function Chat() : JSX.Element {
  const ref : RefObject<HTMLDivElement> = useRef<HTMLDivElement>({ initialValue: null })
  const { messages : Message[], input : string, handleInputChange, handleSubmit, isLoading : boolean, error : Error | undefined } =
    useChat( {api, id, initialMessages, initialInput, sendExtraMessageFields, experimental_onFunctionCall, experimental_onToolCall, streamMode, onResponse, onFinish, onError, credentials, headers, body, generateId}: {
      initialMessages: [
        {
          id: Date.now().toString(),
          role: 'system',
          content: 'Be PennyWise, the financial insights AI. Offer insightful analysis on financial questions, strictly focused on financial tasks. ' +
            'Never provide direct financial advice. Remember to ask questions if necessary for more information. Informed answers only. ' +
            'Stay within the realm of financial analysis and avoid offering actionable guidance.'
        }
      ]
    })
}
```

Fetching GoCardless Token and Banking Institutions:

- First request obtains access token used for subsequent requests
- Access Token is used to get and return valid institutions based on the country code (IE).

```

export async function requestToken(): Promise<any> {
  try {
    const response :Response = await fetch( input: "https://bankaccountdata.gocardless.com/api/v2/token/new/", init: {
      method: "POST",
      headers: {"accept": "application/json"...},
      body: JSON.stringify( value: {
        secret_id: process.env.SECRET_ID,
        secret_key: process.env.SECRET_KEY
      })
    });
    if (!response.ok) {
      console.log("Network response was not ok: ", response);
    }
    return await response.json();
  } catch (error) {
    console.error("There was a problem with your fetch operation:", error);
    throw error; // Rethrow the error to handle it outside the function
  }
}

```

CheezyLeezy1, 10/05/2024, 22:41 • feature/THE-14-17-19-20:

1+ usages 1 CheezyLeezy1

```

export async function getValidInstitutions(accessToken: string) : Promise<any> {
  console.log('accessToken:', accessToken)
  console.log("Authorization:", `Bearer ${accessToken}`);

  try {
    const response :Response = await fetch( input: "https://bankaccountdata.gocardless.com/api/v2/institutions/?country=IE", init: {
      method: "GET",
      headers: {
        "accept": "application/json",
        "Authorization": `Bearer ${accessToken}`
      }
    });
    if (!response.ok) {
      console.log("Network response was not ok: ", response);
    }
    return await response.json();
  }
  catch (e) {
    console.error("There was a problem with your fetch operation:", e);
    throw e; // Rethrow the error to handle it outside the function
  }
}

```

TypeScript serverless function for fetching data from OpenAI API:

- handles incoming POST requests, checks for the presence of a OpenAI API key, and processes provided messages to create chat completions using the model gpt-3.5-turbo.
- function then streams the generated responses back to the client using the OpenAIStream
- errors handled appropriately with regards to current implementation.

```

no usages  ▲ CheezyLeezy1
export const runtime : "edge" = 'edge'
const openai : OpenAI = new OpenAI( {baseUrl, apiKey, organization, project, ...opts}: { apiKey: process.env.OPENAI_API_KEY || '' })

1+ usages  ▲ CheezyLeezy1
export async function POST(req: Request) : Promise<NextResponse<?> | Stre... {
  try {
    if (!process.env.OPENAI_API_KEY) {
      return new NextResponse( body: 'Missing OpenAI API Key.', init: { status: 400 })
    }

    const { messages } = await req.json()
    const response : Stream<OpenAI.Chat.Completions... = await openai.chat.completions.create( body: {
      model: 'gpt-3.5-turbo',
      stream: true,
      messages
    })

    const stream : ReadableStream<any> = OpenAIStream(response)
    return new StreamingTextResponse(stream)
  } catch (error: any) {
    return new NextResponse( body: error.message || 'Something went wrong!', init: {
      status: 500
    })
  }
}

```

Auth Route for GoCardless:

- handles incoming GET requests to obtain bank authentication tokens
- requests token using util function defined above.
- Extracts values from response and sets them as HTTP cookies.
- Security minded function.

```

1+ usages  ± CheezyLeezy1
export async function GET() : Promise<NextResponse<{message:... } {
  const response:BankAuthTokenResponse = await requestToken();
  const accessToken:string = response.access;
  const accessTokenExpiry:number = response.access_expires; // Assuming these properties exist
  const refreshToken:string = response.refresh;
  const refreshTokenExpiry:number = response.refresh_expires;

  cookies().set(
    args: 'authToken',
    accessToken,
    {
      path: '/', // Accessible from all paths
      domain: process.env.NEXT_PUBLIC_DOMAIN || 'localhost',
      httpOnly: true, // Prevent client-side JavaScript access
      secure: true, // Only send over HTTPS connections
      sameSite: 'strict', // Mitigate cross-site request forgery (CSRF) attacks
      maxAge: accessTokenExpiry, // Expires after access token expiry
    }
  );

  cookies().set(
    args: 'authRefreshToken',
    refreshToken,
    {
      path: '/api/auth/refresh', // Only accessible for refresh route (optional)
      httpOnly: true,
      secure: true,
      sameSite: 'strict',
      maxAge: refreshTokenExpiry, // Expires after refresh token expiry
    }
  );

  return NextResponse.json( { body: { message: accessToken }, init: { status: 200 } });
}

```

Transaction Loader:

```

1+ usages  ± CheezyLeezy1
export const TransactionsLoader: React.FC<TransactionsLoaderProps> = ({
  accountNum : string ,
}) => {
  const {
    data: transactionsData : TransactionsData<RevTransactio... ,
    error: transactionsError : Error | null ,
    loading: transactionsLoading : boolean ,
  } = useFetchTransactions(accountNum)
  const {
    data: balanceData : Balance[] | null ,
    error: balanceError : Error | null ,
    loading: balanceLoading : boolean ,
  } = useFetchBalance(accountNum)
  const {
    insights : Insight[] ,
    loading: insightsLoading : boolean ,
    error: insightsError : string | null ,
  } = useFetchInsights(transactionsData)

  if (transactionsLoading || balanceLoading || insightsLoading) {
    return <WaitingScreen statusMessage="Loading data..." />
  }

  if (transactionsError || balanceError || insightsError) {
    return (
      <div className="p-4 text-red-600">
        <h2 className="text-lg font-semibold">An error occurred</h2>
        <p>
          {transactionsError?.message || balanceError?.message || insightsError}
        </p>
        <p>
          Please try refreshing the page or contact support if the issue
          persists.
        </p>
      </div>
    )
  }

  if (!transactionsData || !balanceData) {
    return <div>No data available</div>
  }
}

```

- Utilises custom hooks for fetch insights, balances and transactions.
 - Each being separate API Calls

```

const isRevData : boolean = transactionsData.transactions.booked.every(isRevTransaction)
const isAibData : boolean =
  !isRevData && transactionsData.transactions.booked.every(isAibTransaction)
const isPtsbData : boolean =
  !isRevData &&
  !isAibData &&
  transactionsData.transactions.booked.every(isPtsbTransaction)

return (
  <>
    {isRevData && (
      <Suspense
        fallback=<WaitingScreen statusMessage="Loading transactions..." />
      >
        <DashCards
          transactionsData={
            transactionsData as TransactionsData<RevTransaction>
          }
          balanceData={balanceData}
        />
        <DashMiddle
          transactionsData={
            transactionsData as TransactionsData<RevTransaction>
          }
        />
      </Suspense>
    )}
    {!isRevData && isAibData && (
      <Suspense
        fallback=<WaitingScreen statusMessage="Loading transactions..." />
      >
        <DashCards
          transactionsData={
            transactionsData as TransactionsData<AibTransaction>
          }
          balanceData={balanceData}
        />
      </Suspense>
    )}
  )
)

```

- Then checks the Data against the types of transactions (TypeScript)
- Passes them into a component to be displayed.
- Makes up the Dashboard component, DashCards, DashMiddle and DashAI are the components that receive the data.

```

// Define the start date for the past year
const oneYearAgo = subYears(new Date(), 1)

// Filter transactions to include only those within the past year
const recentTransactions = transactionsData.transactions.booked.filter(
  (transaction) =>
    new Date(transaction.bookingDateTime || transaction.bookingDate) >=
      oneYearAgo
)

// Sort transactions by bookingDateTime or bookingDate in descending order
const sortedTransactions = [...recentTransactions].sort(
  (a, b) =>
    new Date(b.bookingDateTime || b.bookingDate).getTime() -
      new Date(a.bookingDateTime || a.bookingDate).getTime()
)

// Get the latest 5 transactions with identifiable places (creditor or debtor)
const transactionList = sortedTransactions
  .filter((transaction) => getMerchantName(transaction) !== 'Unknown') // Use getMerchantName to filter out unknown places
  .slice(0, 5) // Limit to the first 5 transactions

// Calculate spending statistics
const placeSpendMap: Record<string, number> = {}
const placeCountMap: Record<string, number> = {}

// Iterate through all recent transactions to calculate total spending and count per place
recentTransactions.forEach((transaction) => {
  const placeName = getMerchantName(transaction)
  if (placeName !== 'Unknown') {
    const amount = Math.abs(parseFloat(transaction.transactionAmount.amount))

    // Initialize the maps if the placeName hasn't been encountered yet
    if (!placeSpendMap[placeName]) {
      placeSpendMap[placeName] = 0
      placeCountMap[placeName] = 0
    }

    // Accumulate the total spending and count per place
    placeSpendMap[placeName] += amount
    placeCountMap[placeName] += 1
  }
})

// Convert the placeSpendMap into an array of [placeName, totalAmount, count] and sort
const sortedPlaces = Object.entries(placeSpendMap)
  .map(([place, totalAmount]) => ({
    place,
    totalAmount,
    count: placeCountMap[place],
  })))
  .sort((a, b) => b.totalAmount - a.totalAmount)
  .slice(0, 7) // Take top 7 places

```

- Code to sort the data incoming into a component (DashMiddle)

Code for sorting transactions:


```

export const calculate30DayTotals = <T extends BaseTransaction>(
  data: TransactionsData<T>
): [number, number] => {
  const transactions :T[] = data.transactions.booked

  const today :Date = new Date()

  const startDateCurrent :Date = subDays(today, amount: 30)
  const endDateCurrent :Date = today

  const startDatePrevious :Date = subDays(today, amount: 60)
  const endDatePrevious :Date = startDateCurrent

  const current30DayTransactions :T[] = transactions.filter(
    (transaction :T ) =>
      isAfter(new Date(transaction.bookingDate), startDateCurrent) &&
      isBefore(new Date(transaction.bookingDate), endDateCurrent)
  )

  const previous30DayTransactions :T[] = transactions.filter(
    (transaction :T ) =>
      isAfter(new Date(transaction.bookingDate), startDatePrevious) &&
      isBefore(new Date(transaction.bookingDate), endDatePrevious)
  )

  1+ usages  CheezyLeezy1
  const calculateTotal = (transactions: T[]): number => {
    return transactions.reduce((total :number , transaction :T ) => {
      const amount :number = parseFloat(transaction.transactionAmount.amount)
      return amount < 0 ? total + Math.abs(amount) : total
    }, 0)
  }

  const currentTotal :number = calculateTotal(current30DayTransactions)
  const previousTotal :number = calculateTotal(previous30DayTransactions)

  const percentageChange :number =
    previousTotal !== 0
      ? ((currentTotal - previousTotal) / previousTotal) * 100
      : 0

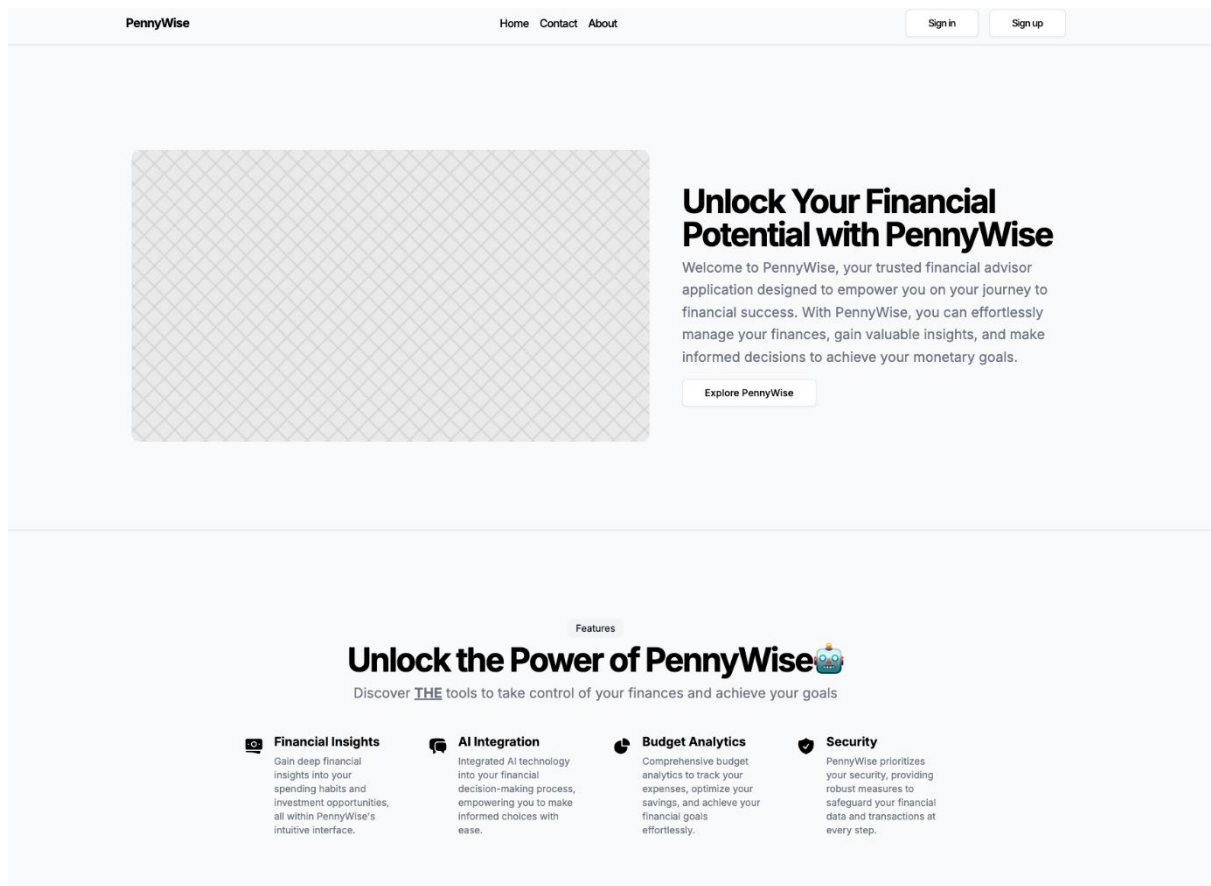
  return [currentTotal, percentageChange]
}

```


2.3. Graphical User Interface (GUI)

Dashboard: Homepage and how it looks, complete with header and footer. All responsive.

- Placeholders Images



Sign in and Sign out Forms – Provided by Kinde with Google Login and Password OTP Functionality:

PennyWise

Register

Get started today!

First name

Last name

Email

Create your account

Or



Continue with Google

Already have an account? [Sign in](#)

PennyWise

Hey friend! Welcome back

Email

Continue

Or



Continue with Google

No account? [Create one](#)

Contact Page:

Contact Support

Fill out the form below and we will get back to you as soon as possible.

First name

Last name

Email

Message

Send message

Contact Support

Fill out the form below and we will get back to you as soon as possible.

First name

First name must be at least 2 characters

Last name

Last name must be at least 2 characters

Email

Invalid email address

Message

Message must be at least 10 characters

- Validation using ZOD Library


Bank Selection UI – Data taken from Banking API and Presented in a UI Component:

PennyWise


[Home](#) [Contact](#) [About](#)

[Log out](#)


Select Your Bank



Allied Irish Banks
BIC: AIBKIE2DXXX



Permanent TSB
BIC: IPBSIE2D



Revolut
BIC: REVOLT21XXX

- User needs to be logged in

Welcome to PennyWise

At PennyWise, we're on a mission to revolutionise the way you manage your finances. Our AI-powered platform empowers individuals like you to take control of your financial future, providing personalized insights, expert advice, and intuitive tools to help you achieve your monetary goals with confidence.

Our Services

Discover an AI Powered Tool designed to meet your financial needs. From budget planning to analytics, PennyWise offers the tools and resources you need to navigate your financial journey with ease.

Our Story


PennyWise was born from a vision to empower individuals to make smarter financial decisions. Founded by a team of passionate innovators, we've dedicated ourselves to creating a platform that combines cutting-edge technology with expert financial knowledge to help you unlock your financial potential.

Our Commitment

- ✓ Innovation
- ✓ Excellence
- ✓ Integrity
- ✓ Empowerment

Contact Us

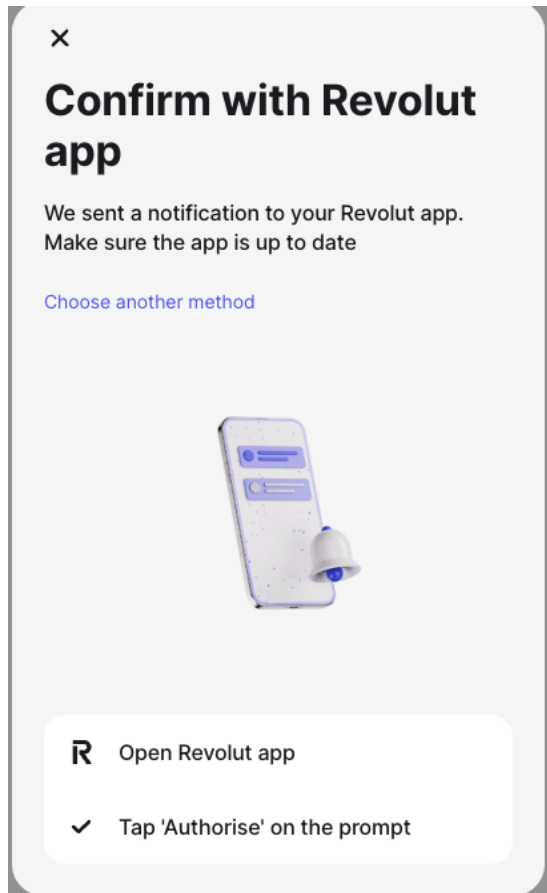
Ready to embark on your journey to financial success? Connect with us today and discover how PennyWise can help you turn your financial dreams into reality.

 info@pennywise.com

 [GitHub](#)

GoCardless:

- Showing Auth Process:



Accounts access request

Authorise **GoCardless** to read your accounts information. Permission will expire on 3 Nov 2024.

Choose accounts

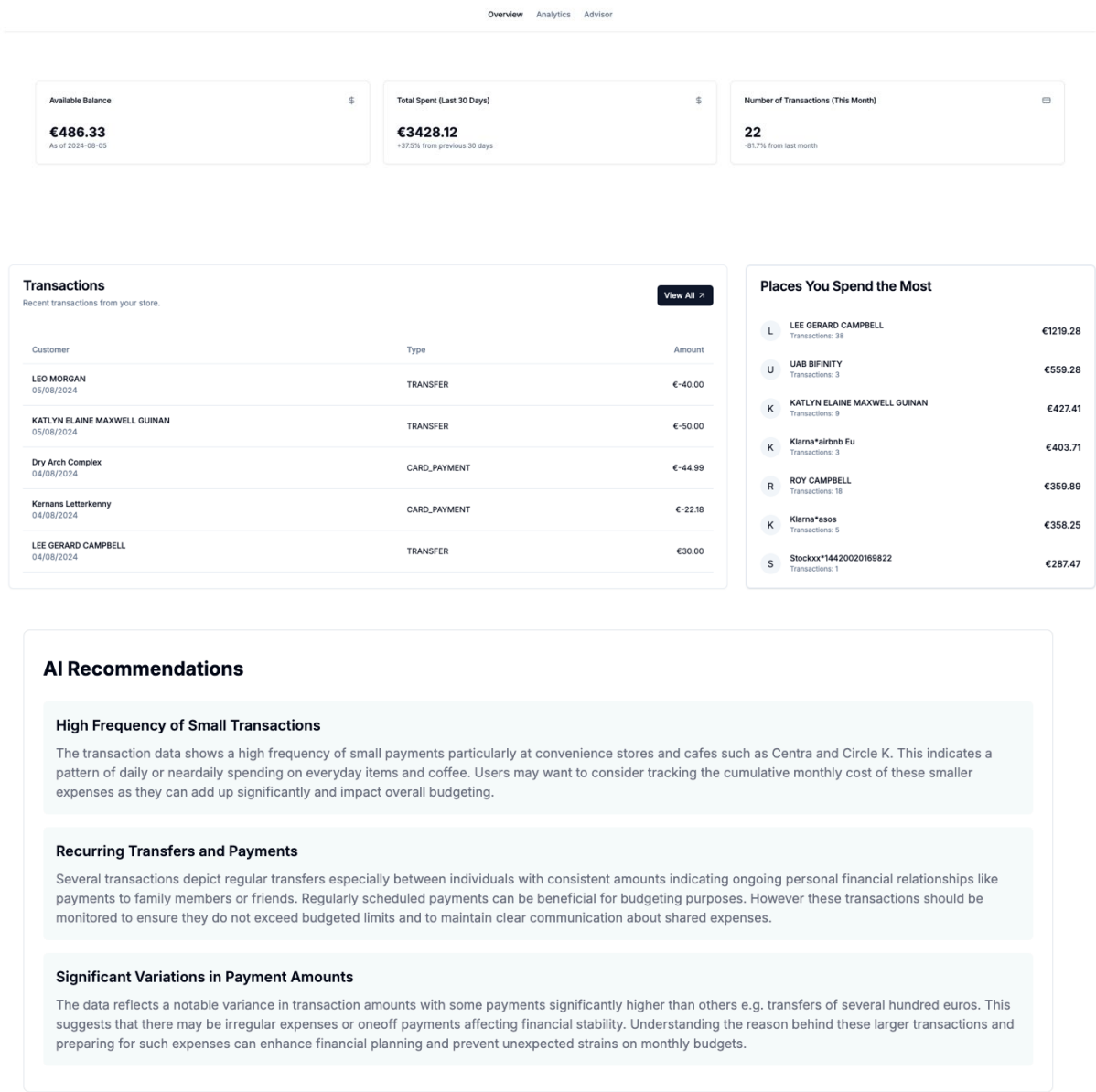


Loading data...

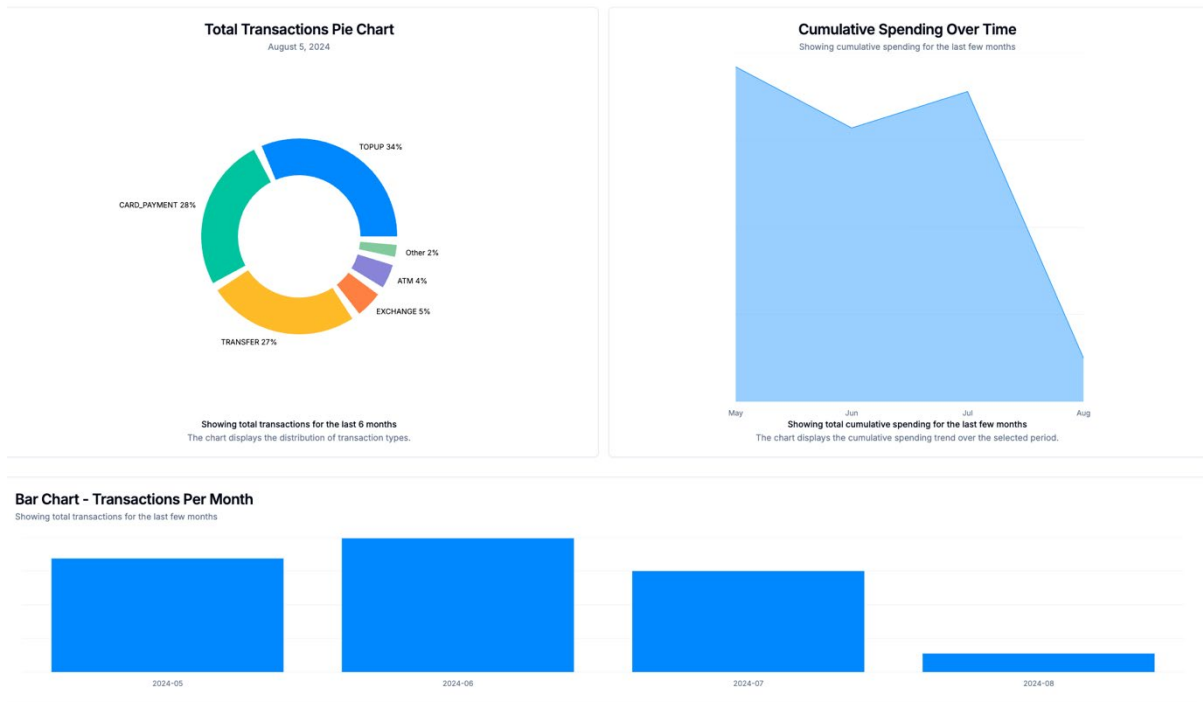
Dashboard:

Showing Data – My own financial data:

Each Section is its own component with different calls and formatting requirements:



Analytics Page:

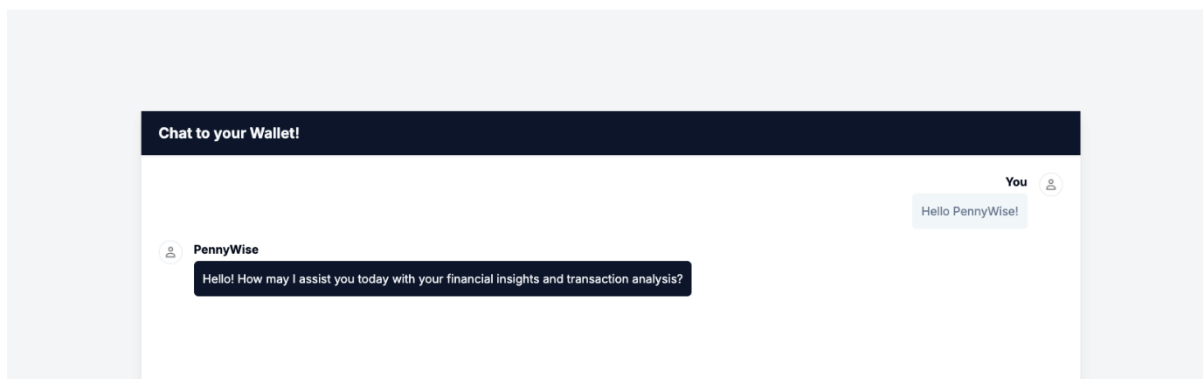


- Interactive
- Shadcn UI charts

AI Chat Component:

Connected with ChatGPT AI – using next API route.

Process-> Give instructions, wait for input, respond.



- Scrollable
- Uses a library to wrap the responses in Markdown, found in a GitHub issue and it worked well for my use case and started researching how to format responses.

2.4. Testing

Testing Tools:

1. Bank Connectivity Testing:
 - Tools: Dummy bank accounts, financial APIs.
2. Manual AI Models Testing for Financial Advisory Topics:
 - Tools: AI models, financial advisory queries.
3. UI Testing using Playwright or Cypress:
 - Tools: Playwright, Cypress.
4. CI/CD Pipeline Before Deployment:
 - Tools: CI/CD pipeline tools (e.g., Jenkins, GitLab CI/CD).

Testing Plan:

1. Bank Connectivity Testing:
 - Scenario: Simulate transactions and account activities using dummy bank accounts.
 - Evaluation Criteria: Ensure seamless integration with financial institutions and transaction handling.
 - Testing Approach: Conduct end-to-end system tests focusing on reliability and connectivity consistency.
2. AI Models Testing with Sample Spending Spreadsheets:
 - Scenario: Input financial simulated data and evaluate AI-driven insights.
 - Evaluation Criteria: Assess accuracy and relevance of AI-generated financial advice.
 - Testing Approach: Systematically test different spending scenarios and refine models based on results.
3. AI Models Testing for Financial Advisory Topics:
 - Scenario: Generate financial queries and evaluate AI responses.
 - Evaluation Criteria: Measure relevance and helpfulness of AI-generated financial advice.
 - Testing Approach: Create diverse queries, analyse responses, and refine the model.
4. UI Testing using Playwright or Cypress:
 - Scenario: Simulate user interactions with the interface.

- Evaluation Criteria: Ensure smooth user experience, identify UI issues.
 - Testing Approach: Implement automated UI tests to test responsiveness and functionality.
5. CI/CD Pipeline Before Deployment:
- Scenario: Automate build, testing, and deployment processes.
 - Evaluation Criteria: Assess efficiency in catching errors and facilitating deployments.
 - Testing Approach: Implement continuous integration and delivery processes, conduct automated tests.

2.5. Evaluation

- The model with the promise was ChatGPT 4o Mini – less cost in tokens and returned data the way I expected it to.
 - This model does not hallucinate as much as others and sticks to task well.
- UI Testing Completed With Playwright for pages that are accessible without security tokens.
 - Cannot mimic login like other testing frameworks as to not expose sensitive data.
- CI/CD Pipeline was made easy using a combination of Vercel and GitHub.
 - This included staging environments
 - Prod environments
 - Automated Linting, Formatting and Running of Tests
- Bank Connectivity Testing took some time due to the complex nature of how data was returned, each bank kept a similar structure but with subtle differences.

3.0. Conclusions

Advantages:

1. **Personalised Financial Guidance:** Tailored Financial Insights and Guidance for people who may be less financial inclined, being able to talk to wallet seems like a utility.
2. **Integrated AI Models:** Uses an integrated AI model to provide these insights – since AI is a buzzword with regards to start-up, and the amount of data a AI model can handle, this is both time saving and promotable.
3. **Financial Management:** Platform for users to manage financial accounts and access financial information and tailored insights.
4. **Clean UI:** Clean and simple Interface built with Modern Web Development Infrastructure.

Disadvantages:

1. **Dependency on External APIs:** Relies on external APIs for fetching financial data, which introduces points of failure for the application.

2. **Privacy Concerns:** Sensitive nature of financial data would require a large scale security overhaul and security experts if the application was actually being built for wide-scale usage.
3. **Limited Customisation:** Lacks flexibility in regards to user preferences and financial goals.

Strengths:

1. **Scalable Architecture:** Built on a server-side rendered architecture, scalable, with regards to Vercel who manage the scalability of applications deployed there.
2. **Secure Authentication:** Implements server-side authentication and session management for enhanced security, limits client side vulnerabilities.
3. **No Access to Data:** Users of the site have a token that expires and with it, the ability to test with your data.
 - Being a Developer, I have a no access to the data from any real users.
4. **Modern UI:** Built with modernity in mind and the latest tools for web development.

Limitations:.

1. **Complex Integration:** Integrating with multiple APIs may prove complex and time-consuming, could require ongoing maintenance and updates.
2. **Limited AI Capabilities:** While AI-driven insights are valuable, they may be limited in scope and accuracy – may be difficult to refine in current implementation.
 - Going forward training a custom model would be better suited for this use case.

3.0 Further Development or Research

1. **Training my own AI Model:** training a proprietary AI model tailored specifically for financial analytics would be the main focus of further development. This would involve getting largescale datasets and supervised and unsupervised learning, LLaMA models provided by Meta show the most promise in respects to this.
2. **Exploring Different Ways of Offering Financial Analytics:** Research into more innovative approaches for presentation of financial analytics to users. Predictive modelling or visualisations.
3. **More AI Functionalities:** Expanding AI functionalities of the application which could summarization techniques or more advanced insights generation.
4. **Mobile Application:** Development of a mobile application version of the PennyWise platform could significantly expand its reach if taken serious.
5. **Integration with Existing Applications:** Offering APIs or integration modules, PennyWise could seamlessly integrate with popular banking apps, budgeting software, or personal finance platforms.

4.0 References

Build and deploy the best web experiences with the frontend cloud (no date) Vercel.
Available at: <https://vercel.com/home> (Accessed: 12 May 2024).

Group, P.G.D. (2024) PostgreSQL. Available at: <https://www.postgresql.org/> (Accessed: 12 May 2024).

Simplify working and interacting with databases (no date) Prisma. Available at: <https://www.prisma.io/> (Accessed: 12 May 2024).

Typescript-first schema validation with static type inference (no date) TypeScript-first schema validation with static type inference. Available at: <https://zod.dev/> (Accessed: 12 May 2024).

Next.js by vercel - the REACT framework (no date) Next.js by Vercel - The React Framework.
Available at: <https://nextjs.org/> (Accessed: 12 May 2024).

<https://claude.ai/>

5.0 Appendices

This section should contain information that is supplementary to the main body of the report.

5.1. Project Proposal

Contents

1.0 Objectives	2
1.1 Background	3
2.0 State of the Art	4
3.0 Technical Approach	5
4.0 Technical Details	6
5.0 Special Resources Required	8
6.0 Project Plan	8
7.0 Testing	11

1.0 Objectives

The PennyWise project's objectives are focused, clear, and serve as quantifiable targets that guide the development of the application and to reach the intended results. These objectives, which reflect practical deliverables and benchmarks, are separate from the higher-level project

goals, as stated in the clarification on project objectives found here: (<https://asana.com/resources/how-project-objectives>).

Among these goals are:

Bank Connectivity and Secure API Implementation:

- Successfully integrate GoCardless to fetch any transaction data securely within a specified timeframe.
- Ensure consistent and dependable connectivity, allowing users to access their financial information via the PennyWise app.
- Integrations with at least two banks.

Development of Security Infrastructure:

- Use strong encryption and authentication mechanisms to protect sensitive financial data.
- Establish a secure key storage mechanism throughout the project to prevent unauthorised access and to preserve user privacy.
- Establish a secure and fail-safe way to allow users to trust within the applications security protocols and limit sensitive data exposure.

Backend Development:

- Utilise Docker for containerisation and compatibility with the selected programming languages.
- Implement Docker containers with TypeScript for the back-end, enhancing code maintainability and developer productivity.
- Possible Java or Python Integration dependent on further research.

Front-end Development:

- Based on research and user experience needs, select and deploy the best front-end framework.
- Create an intuitive and visually appealing user interface that matches with the project's goals and improves overall usability.

AI Model Integration:

- For personalised financial advice, use pre-trained AI models, with an emphasis on continual reinforcement training to improve accuracy.
- Implement a system for updating AI models on a regular basis in order to respond to changing financial trends and user demands.

Other Objectives:

- Testing and debugging: Perform extensive testing to discover and resolve issues, delivering a dependable and error-free user experience.
- Validate the efficacy of encryption technologies and authentication systems through security testing.
- Deployment & Launch: Carry out a deployment of the PennyWise application, ensuring that all components are seamlessly integrated.
- Launch the application to the public, completing the project's objectives and providing a AI-based financial tool.
- Establish systems for continual monitoring and improvement of the application's performance, user input, and security measures.
- Make plans to update and improve your website on a regular basis depending on user input and developing financial trends.

1.1 Background

The PennyWise project was inspired by a thorough assessment of the current AI startup ecosystem, which revealed a significant need in complete financial management solutions. The observation of ChatGPT releasing custom command or templated AI agents spurred the concept for a pioneering "Talk to Your Wallet" application, amidst the proliferation of AI applications mostly working as ChatGPT wrappers.

PennyWise intends to give customers intelligent advice and personalised money management techniques, similar to how diagnostics for cars can identify problems. The initiative takes on a new relevance in this context of the rising cost of living issue, which is particularly evident in places such as Dublin. The objective is to develop an app that not only educates but also empowers people to manage their finances.

The chosen technology stack, which includes a combination of Java and TypeScript for the backend and Angular for the frontend, provides a twofold benefit. On the one hand, it enables skill development inside the stack which is used within my professional work and life. On the other hand, it makes use of languages that are trusted by large-scale institutions, resulting in a solid and scalable design. The use of Docker improves the project's efficiency and compatibility.

Though seemingly difficult, integrating AI models is an intriguing challenge. The initial phase involves utilising APIs to customise the models to the demands of the project, opening the path for further training and the absorption of critical vectored information. This method not only corresponds with the project's aims, but it also provides a vital learning opportunity in the field of AI development.

Cybersecurity is a critical component of the PennyWise initiative. The crucial relevance of cybersecurity within this project underscores the commitment from me, the developer to understanding and implementing comprehensive cybersecurity measures. Access to lecturers and materials will be a valuable tool in learning cybersecurity procedures, assuring the application's resilience in the face of potential threats.

PennyWise emerges as a project anchored on meeting a real-world need in the middle of an ever-changing field of AI applications. The project's comprehensive approach is aided by the technological stack chosen, the integration of AI models, and a significant emphasis on cybersecurity.

2.0 State of the Art

PennyWise sees itself as the "one tool to rule them all" in financial management, providing a game-changing solution that goes beyond the constraints of existing applications. The ability to effortlessly interface with any bank within Ireland is at the heart of its distinctiveness, offering consumers with unprecedented access and control over their financial data and to talk to a financial advisor through the clicking of a few buttons.

What sets PennyWise apart is its distinctive focus on integrating artificial intelligence, through advanced language models like ChatGPT, to offer personalised and dynamic on the fly financial advice. While existing applications may provide static budgeting and tracking features, PennyWise aims to enhance the user experience by allowing individuals to engage in meaningful conversations with their own financial advisor.

Similar Applications:

Revolut – Offers financial insights, subscription tracking, budget or cost alerts and charting to allow users to better visualise spending.

MeetCleo – allows users to upload a payslip, which can then create budgeting, bill planning, overdraft alerts, spending tracking.

Wally.me – Allows user to utilise connectivity with a bank and follow a similar approach compared to my application, users can ask questions about specific transactions, spending habits and ask how best to budget and save.

My Application:

PennyWise distinguishes itself as a unique solution in the Irish market. Unlike other applications, comparable tools have received less attention, and PennyWise intends to fill the hole by offering a specialised solution for the Irish market. The lack of a comparable application in Ireland highlights PennyWise's unique character, which not only integrates current capabilities but also pioneers the integration of powerful AI models for unrivalled user engagement

Pennywise aims to be more than just a “wrapper” for more large scale models. PennyWise aims to be “the” personal adviser application existing within the Irish market. A major benefit is allowing people to build before you, and try to rectify any shortcomings and critiques that may be found.

3.0 Technical Approach

1. Task Generation:

- Daily Planning: Set aside one day every week to plan out small, doable projects for the next week. Divide features and functions into discrete tasks that can be performed in a week.
- Prioritise activities based on their dependencies, significance, and overall project goals. Ascertain that key tasks correspond to project milestones.
- Detailed User Stories: Supplement your task descriptions with extensive user stories that outline the user's point of view as well as the intended outcome of each activity.

2. Kanban Development:

- Use Atlassian or Trello to create a Kanban board where all tickets and work can go. Columns for To Do, In Progress and Done will be visible to provide a clear overview.
- Set Work in Progress Limits to prevent overload and maintain a steady flow of work.
- Leave comments on stories about how difficult or easy the previous story was, this will allow me make adjustments going forward to the my Kanban process and provide a route for continuous improvement.

3. Milestone Planning:

- Strategic Milestones: Determine strategic milestones that are in line with the overall project objectives. Milestones might include successful GoCardless integration, completion of AI model integration, or the first user testing phase.
- Timeline Adjustments: Regularly review your project timeline and milestones. Changes may be required as a result of progress, difficulties faced, or comments received.

4. Continuous Documentation:

- Documenting progress, decisions and any challenges faced – this documentation will serve as a tool for fully comprehending the things I will do, and why – and will provide a knowledge base for looking back on in the future.

5. Consistent and Thorough Testing:

- Including testing milestones, even if they are tiny. Document any feedback to confirm assumptions and ensure the applications meets expectations.
- Make time for comprehensive testing and troubleshooting. Putting my application's dependability and stability first.

- Include security audits in my milestone planning. Ensuring that security precautions are built into the development process.

6. Time Management:

- Allocate specific time blocks for development, testing, and documentation. Effective time management is crucial for balancing various aspects of the project.
- Use the resources provided by the college for effective time management and creating work schedules.

4.0 Technical Details

Java:

- Backend Development: Java is renowned for its reliability, scalability, and security. It's well-suited for building the backend infrastructure, handling data processing, and for integrations with financial institutions.

Frontend Development(Angular, TypeScript):

- Angular, a powerful frontend framework for building dynamic, single-page web applications.
- TypeScript, a superset of JavaScript, brings static typing and modern language features. It's ideal for developing a dynamic and responsive frontend, and is the underlying language used with Angular, ensuring a smooth user experience.

Spring Boot (Java):

- Purpose: A widely-used framework for building robust Java-based, microservices-driven applications.
- Benefits: Offers functionality out of the box like security, dependency injection, and integrations with other Spring projects, facilitating rapid development.

Docker:

- Purpose: For containerization, enabling consistent deployment across different environments.
- Benefits: Enhances scalability, isolation, and simplifies deployment and testing processes. Allows for simple testing in a secure environment.

GoCardless API:

- Purpose: Integrating with financial institutions to fetch transaction data securely.
- Benefits: An API that simplifies access to bank transaction data, supports secure and seamless integrations.

Possible:

Hugging Face Transformers (AI Models):

- Purpose: Pre-trained language models for natural language processing and understanding.

- Benefit: Easy integration of powerful ai capabilities that run offline or locally. Can be easily trained on user data.

Important Algorithms and Approaches:

1. Encryption Algorithms:

- Purpose: Protect sensitive financial data both in transit and at rest.
- Approach: Implement industry-standard encryption algorithms (example: AES-256) to ensure the highest level of security.

2. Least Trust Principles:

- Purpose: Minimise trust assumptions within the system to enhance security.
- Approach: Granting only the minimum access necessary for each component or user.

3. CIA Triad (Confidentiality, Integrity, Availability):

- Purpose: Ensure the core principles of information security.
- Approach: Implement measures such as access controls, data validation, and redundancy to safeguard confidentiality, integrity, and availability.

4. MVC (Model-View-Controller) (Possible):

- Purpose: Provide a structured architecture for the frontend application.
- Approach: Implement a separation of concerns, with the model representing the data, the view managing the user interface, and the controller handling user input and system events.

5. Singleton Pattern (Where Applicable):

- Purpose: Ensure a single instance of a class, particularly useful for managing shared resources.
- Approach: Implement a singleton pattern for components that could or may have a single instance throughout the application, optimising resource usage.

6. Microservices Architecture (Possible):

- Purpose: Divide the application into small, independent deployable services.
- Approach: Considering a possible microservice architecture for the differing AI functionalities, ensuring modularity and scalability. Each microservice can focus on specific AI tasks, promoting maintainability.

5.0 Special Resources Required

1. Angular Documentation (angular.dev):

- Purpose: Angular is a key framework for the frontend development of PennyWise. The new official Angular documentation provides comprehensive guides, tutorials, and references necessary for building user interfaces.

2. TypeScript Resources (Total TypeScript Course by Matt Pocock):

- Purpose: TypeScript is the chosen language for frontend development. The "Total TypeScript Course" by Matt Pocock is a valuable resource for in-depth learning, covering TypeScript fundamentals, advanced features, and best practices. I have started this course and plan to finish before beginning my project.

3. YouTube for Security Implementations:

- Purpose: Security is a critical aspect of PennyWise. YouTube tutorials and educational channels focused on security implementations, encryption methods, and best practices will provide practical insights and guidance.

4. School Lecturers:

- Purpose: Use the expertise of school lecturers for guidance, mentorship, and academic insights

5. Baeldung for Java (if used):

- Purpose: Baeldung is a well-known resource for Java developers, offering tutorials, articles, and best practices. If Java is used in the backend development of PennyWise, Baeldung will serve as a reference for Java-related implementations.

6.0 Project Plan

Project Timeline:

Start Date: 19 Dec 2023 Target Completion: End of July 2024

Milestones:

1. Bank Connectivity and Secure API Implementation (Estimated Completion: Jan 2024)

- Tasks:
- Research and select financial institutions for initial integration.
- Set up and test connections with GoCardless API.
- Implement secure transaction handling.

2. Security Infrastructure Development (Estimated Completion: Feb 2024)

- Tasks:

- Implement robust encryption algorithms for data at rest and in transit.
- Develop secure authentication mechanisms.
- Establish key storage procedures.

3. AI Model Integration (Estimated Completion: March 2024)

- Tasks:
- Research and choose pre-trained AI models for financial insights (e.g., Hugging Face Transformers).
- Large Scale Model Wrappers
- Research prompt training, input training and limiting malicious input.
- Integrate chosen AI models into the application architecture.
- Conduct initial testing of AI-driven features.

4. Front-end Development (Estimated Completion: April-May 2024)

- Tasks:
- Investigate Angular and TypeScript for the front-end.
- Choose the most suitable framework based on research and requirements.
- Develop the user interface with a focus on user experience.
- Make Frontend Secure and implement measures in line with modern software applications according to NIST.

5. Back-end Development (Estimated Completion: April-May 2024)

- Tasks:
- Docker for containerization during development.
- Implement backend functionality using Java and TypeScript.
- Ensure seamless integration with the front-end and AI components.

6. Testing and Debugging (Estimated Completion: Jun 2024)

- Tasks:
- Conduct rigorous testing to identify and rectify issues.
- Perform security testing to validate encryption and authentication effectiveness.
- Iteratively debug and refine the application.

7. Application Refinement (Estimated Completion: Jul 2024)

- Tasks:
- Gather user feedback on the application's usability.
- Make iterative improvements to the user interface.
- Ensure the application meets user experience standards.

8. Deployment and Launch (Estimated Completion: End of July 2024)

- Tasks:
- Prepare the application for deployment.
- Gather Documentation
- Conduct final testing in a production environment.
- Officially launch PennyWise for public use.

9. Continuous Monitoring and Improvement (Jun-Ongoing)

- Tasks:
- Establish mechanisms for monitoring application performance.
- Implement user feedback loops for continuous improvement.
- Plan for future updates and enhancements.

Resource Allocation:

- Tools and Platforms:
- Angular, TypeScript
- Java
- Docker
- GoCardless API
- AI Models

Risk Management:

- Identified Risks:
- Integration challenges with financial institutions.

- Potential security vulnerabilities.
- User interface design not aligning with user expectations.
- Mitigation Strategies:
- Regular security audits and updates.
- Continuous testing and feedback loops for application refinement.

Monitoring and Evaluation:

- Testing Sessions:
- Bi-weekly user testing sessions for feedback and improvement.
- Midpoint Documentation:
- Detailed documentation and analysis at the project midpoint for refinement.

7.0 Testing

1. Bank Connectivity Testing:

- Scenario: Simulate transactions and account activities using dummy bank accounts.
- Evaluation Criteria: Ensure seamless integration with the chosen financial institutions, transaction fetching, and data handling.
- Testing Approach: Conduct end-to-end system tests, focusing on reliability and the consistency of the bank connectivity.

2. AI Models Testing with Sample Spending Spreadsheets:

- Scenario: Input financial simulated data into the application and evaluate AI-driven insights and recommendations.
- Evaluation Criteria: Assess the accuracy and relevance of financial advice generated by AI models based on simulated spending patterns.
- Testing Approach: Systematically test different spending scenarios, measure AI responses, and refine models based on the results.

3. AI Models Testing for Financial Advisory Topics:

- Scenario: Generate queries related to financial advisory topics and evaluate AI responses.
- Evaluation Criteria: Measure the relevance, and helpfulness of AI-generated financial advice. Try to limit hallucinations.
- Testing Approach: Create a diverse set of financial queries, analyse the AI models responses, and refine the model for improved advisory capabilities.

4. UI Testing using Playwright or Cypress:

- Scenario: Simulate user interactions with the application interface.
- Evaluation Criteria: Ensure a smooth and intuitive user experience, identify and fix any UI issues.
- Testing Approach: Implement automated UI tests using tools like Playwright or Cypress to test the responsiveness and functionality of the UI.

5. CI/CD Pipeline Before Deployment:

- Scenario: Automate build, testing, and deployment processes using a CI/CD pipeline.
- Evaluation Criteria: Assess the efficiency of the CI/CD pipeline in catching errors early and facilitating deployments.
- Testing Approach: Implement continuous integration and delivery processes and conduct automated tests at each stage of the pipeline.

6. Routes Testing, UI Testing, Security Testing, Zap Scanning:

- Scenario: Test application routes, UI components, and security measures.
- Evaluation Criteria: Ensure the correctness of routes, the resilience of UI components, and the effectiveness of security implementations.
- Testing Approach: Conduct route testing, UI testing, and use tools like Zap Scanning for security assessments. Address any vulnerabilities identified.

5.2. Reflective Journals

6.0 Supervision & Reflection Template

Student Name	Lee Campbell
Student Number	X20115075
Course	Computing, Final Project, Fourth Year
Supervisor	Enda Stafford

7.0

8.0 Month:

What?

Reflect on what has happened in your project this month?

This month, was an important month for the Project. I had to think of an idea, create a written and video based pitch. The video pitch was made by me, because I felt I needed something to put the idea in my mind onto "paper" and document it. The idea for my project is a talk to your wallet application, where you essentially have your own personal financial advisor and something you can use to talk to your wallet and money and see where you can make savings. I thought of this idea since myself and my sisters in my family seem to have a real issue with "pacing" your money out over a month and seemingly buying into things

without much thought. I personally think this is an issue with how all money in today's age is essentially digitised and we do not get to feel the money in our hands, with new applications and fintech like Revolut and PayPal for example.

For a project idea to be worthy, in my head, it has to solve a problem and have a possible start-up style to it, because I think its important to understand development fully, from idea to execution and delivery.

I completed my project pitch and uploaded it, and seemingly Enda Stafford, my Computing Project Lecturer, seemingly liked it – which pushed me to pursue it even more. I think it always helps when someone sees your version and adds some positive feedback. I also thought of a cool name, PennyWise.

I research a number of applicable resources that I can use for my project, but at this stage I am not really sure what's needed, since I haven't done a full-stack application build before. It's hard to discern what's over-development or what is actually needed. I have found a website, GoCardless, which allows you to essentially import your account from a bank, create an api and query it. So for each customer that uses my application, I will try and implement this.

However I will need to complete a Proof of Concept on this first to see if I can even do it. As it needs to be agnostic and work for each user that interacts.

Another Proof of Concept I need is testing a number of AI's through APIs and see how I can use models. I was thinking of training my own on Budgeting/Financing Books but seems like more work than it is worth. I will research this further though.

In terms of languages and frameworks I will use, it will probably be a full-stack application, using Node, TypeScript, Next and React. With some database connectivity and strong security. I will reach out to my security fundamentals teacher for help with this task.

So What?

Consider what that meant for your project progress. What were your successes? What challenges still remain?

Challenges that remain are implementing the Proof of Concepts (POCs) mentioned before, and trying to navigate what is over-development and essential for development. The overall challenges that remain are actually finding and splitting my time to build the application. I am currently pursuing an Azure Fundamentals and AWS Developer Associate Certificates. These can aid my project if I needed to implement some AI Model training or use hosting provide my these services. My worry as always, is time management, since I am completing certifications, completing other modules applications and CA's and trying not to fall behind in work, my current work-life is quite stressful.

So challenge one is time management, and I would say a close secondary is understanding and implementing what's needed for the task. A big worry is getting halfway or close to an end and realising it cannot be done how I initially envisioned.

My successes were coming up with the idea, completing the written and video-based pitch, which essentially created a visualisation of what is needed to complete my project and it has given me a vision. On the idea front, I am extremely happy about the idea, how unique it is, how people react when I tell them, and how it seemingly hasn't been solved currently.

Now What?

What can you do to address outstanding challenges?

Proper Time Management Schedule

Talk with more experienced Developers on the Implementation Front

Complete Proof of Concepts and try to understand them deeply.

Research and complete write ups on the benefits of certain tools over another

Make sure I understand each aspect of the project

Make the project into Deliverables and follow an MVP (Minimum-Viable-Product) Approach to Development

Student Signature

Lee Campbell

Supervision & Reflection Template

Student Name	Lee Campbell
Student Number	X20115075
Course	Computing – 4 th Year – Part Time

Supervisor	Shivani Jaswal
-------------------	----------------

Month: November

What?

This month, my focus was primarily on meeting college deadlines, which significantly limited my ability to make progress on my project. Despite this, I achieved two notable certifications: Azure Fundamentals and Java IT Specialist through Pearson Vue. While these accomplishments are significant, they consumed much of my time, leaving me feeling somewhat unproductive and frustrated about not advancing my project as planned.

In the moments I managed to dedicate to the project, I developed a basic conceptual framework for the application's architecture. Initially, I considered a serverless approach, but a late-night reflection led me to lean towards a monolithic structure. This idea was inspired by the banking sector, which often favors monolithic systems for their proven reliability, despite not being the most cutting-edge technology.

Ethical considerations also came into play, particularly regarding the application's reliance on user integration. I've decided to use my personal banking information from Revolut and Permanent TSB as test data, ensuring the project's initial testing phase is both practical and ethically sound.

During my initial meeting with Shivani, my supervisor, we discussed the project's technical aspects. We decided that the primary programming languages would be TypeScript and Angular, covering both front-end and back-end development. This decision opens up new avenues for exploration, particularly in areas like Node.js, Express, Mono-Repos, and Bun, a runtime bundler I'm eager to experiment with.

So What?

The completion of the two certifications, while time-consuming, has undoubtedly added to my skill set, potentially benefiting the project in the long run. The decision to adopt a monolithic architecture marks a significant step in defining the project's direction. It reflects a thoughtful consideration of stability and reliability over trendier technological choices.

The ethical decision to use my own banking data for initial testing underlines my commitment to responsible development. It also simplifies the initial testing phase, allowing for a more controlled and personal understanding of the application's functionality. I am hoping to try and extend the application to use any end-users data, however this needs to be discussed more with Shivani at a later date.

However, the challenge of balancing academic responsibilities with project development remains. My limited progress this month highlights the need for a more effective time management strategy, especially as I delve into new technologies like Angular and Node.js. The one benefit from all this, was my Lecturer, Enda Stafford, gave a great lecture last week on the importance of time management, understanding how to plan effectively will be essential as the project progresses.

Now What?

Moving forward, my immediate goal is to deepen my understanding of Angular, which is crucial for the project's development. I plan to create diagrams and detailed plans to better visualise the application's structure and implementation strategy. This has a two-fold benefit, it will add to my comprehension of what I am developing, it will also provide a clear roadmap for development and allow me to better break down my work tasks.

The upcoming Christmas break presents an ideal opportunity to get started with the actual development and research phase of the project. I intend to utilise any free time I have to overcome the setbacks experienced this month and make some substantial progress.

By immersing myself in new technologies and dedicating focused time to development, I aim to address the current challenges and move the project forward significantly.

Student Signature

Lee Campbell

Supervision & Reflection Template

Student Name	Lee Campbell
Student Number	X20115075
Course	Computing – 4 th Year – Part Time
Supervisor	Shivani Jaswal

Month: December

What?

This month, the focus was primarily on finishing studies and preparing for tests, having decided to take a break from my Azure and Java Certifications, this month was more about research, researching the different tools I could use, basic principles of what I am building and how best to go out about.

I also began researching about how I could implement proper security procedures within the application and how best to go about adding users, the overall design and how I should make my project look. This ultimately concluded with a better mental framework of what I was doing. I did a small Angular course, tour of heroes and watched some videos of new features added in Angular 17.

So What?

Since I now know what I am doing, how it should look, it's just about making that jump and to begin building it, having decided December was more of a mental break, something I had not anticipated last month, but

the constant study and programming while working in a software based role, has made me to decide to take a break, as I have struggled with meeting deadlines and have begun to feel the allure of procrastination.

Now What?

Moving forward, January will be the official start date, where I will begin building the project, I hope to no longer feel the burnout and began actual development, as the process and tools I am using is all well-defined and I can picture in my head what I need to do.

Student Signature

Lee Campbell

Supervision & Reflection Template

Student Name	Lee Campbell
Student Number	X20115075
Course	Computing – 4 th Year
Supervisor	Shivani Jaswal

Month: January

What?

Reflect on what has happened in your project this month?

In January, the journey began with the actual development and setup of the Angular 17 project. The foundational steps included setting up a GitHub repository and creating a Kanban Board within it. The decision-making process led to the selection of Angular 17, a technology familiar to my team, to streamline development. A key milestone was the completion of the project setup, which included efficient bundling, code quality linting, and the implementation of pre-commit hooks and formatting tools.

So What?

Consider what that meant for your project progress. What were your successes? What challenges still remain?

This month was more than just technical setups; it was a deliberate effort to lay the groundwork for a strong and collaborative development environment. The decision to use Angular 17, despite its unfamiliarity, was motivated by a need for being practical and attempting to develop a modular, scalable and strict architecture. Every step, from bundling to linting, was designed to maintain high code standards and ensure a smooth development process.

Challenges that remain include picking an AI Model, possibly picking an Algorithm or developing a complete process for linking up with the service that provides banking details. Also, keeping on top of work and college commitments with other subjects.

Now What?

What can you do to address outstanding challenges?

Looking ahead, the groundwork laid in January provides a solid foundation for the journey ahead. The reflection entails more than just completing tasks; it also includes considering how these decisions set the tone for the entire project. The emphasis now is on keeping up the progress, the work/life balance at the moment within my life is difficult, as I am also undertaking a exam for AWS Developer Associate, which requires a lot of attention, on top of trying to balance personal commitments and my own work and career. Ultimately I know I can continue and complete this project, I just need to implement proper time management and at least get a MVP up and running with the coming months

Student Signature

Lee Campbell

Supervision & Reflection Template

Student Name	Lee Campbell
Student Number	X20115075
Course	Computing – 4 th Year
Supervisor	Shivani Jaswal

Month: February

What?

Reflect on what has happened in your project this month?

February marked a shift towards frontend development, as well as the realisation that React could have been a more straightforward option. However, the benefits of learning Angular 17 remained since my team in work use it and I do not want to fall behind with our tech stack. The month saw the completion of a visually appealing Landing Page with animations designed in a bento-style layout. Since the application will only have me as a User, and I wanted to complete a comprehensive web application, I decided I would add the front end components for logging in and signing up. Just to make it somewhat complete.

So What?

Consider what that meant for your project progress. What were your successes? What challenges still remain?

Despite the perceived ease of React, the decision to stick with Angular 17 demonstrates a personal commitment to growth and a willingness to take on new challenges in my opinion. The finished Landing Page, with an emphasis placed on modern design and engaging features, reflects my desire for a visually appealing and user-centered frontend. Creating authentication pages, even for personal use, is me demonstrating a commitment to providing a comprehensive web application experience.

Overall I was happy with the progress I made this month, it was quite difficult to actually begin and get down to it so this was a huge hurdle to overcome, thankfully there is plenty of time left and I plan to incorporate a number of hours a week to development.

Challenges that remain are designing the forms with Angular's new form functionality and features.

Now What?

What can you do to address outstanding challenges?

To address existing challenges:

Investing in Education:

I enrolled in a course covering the new features of Angular 17. This move is an attempt to broaden my comprehension and possibly offer solutions for resolving issues with form functionalities.

Reading Documentation:

Reading the angular.dev new docs for any pointers or help. My understanding of Angular 17's features will improve as a result, facilitating problem-solving and development.

Effective Time Management:

I find that blocking off time on my calendar for extended periods of deep work helps me face obstacles with concentration and productivity.

Student Signature

Lee Campbell

Supervision & Reflection Template

Student Name	Lee Campbell
Student Number	X20115075
Course	Computing – 4 th Year
Supervisor	Shivani Jaswal

Month: March

What?

Reflect on what has happened in your project this month?

March started on a more solemn note, unfortunately I ran into some issues with Angular v17, and with the lacking documentation online, and this not being my knowledge base. I made the decision to jump into Next and React for building out my project. React and Next being extremely well-maintained, used throughout the industry and well documented, it was a massive mistake to begin with a language I was not well informed on. This has made me doubt the deadline, I am feeling a level of dread having wasted so much time.

So What?

Consider what that meant for your project progress. What were your successes? What challenges still remain?

“Despite the perceived ease of React, the decision to stick with Angular 17 demonstrates a personal commitment to growth and a willingness to take on new challenges in my opinion”. My quote from last month it this second, however I think actually delivering on what I have said I would, and my knowledge base on React, I should have went with this in the first place. In terms of progress, I have begun again, and have completed the landing, contact and about page over a weekend. I was determined to build as much as possible, I first begun my taking the NextJS course on vercel, which gave me the foundation to begin building, I studied different UI libraries, how people build things with TailWind, and begin building immediately. This was a huge success. Challenges remain and I am worried about progressing into the dashboard/bank integration part, as this will be the biggest hurdle, however I have a picture of the UI component in my head of how it should look like. Adding Authorisation may also prove difficult.

Now What?

What can you do to address outstanding challenges?	
<p>To address these challenges, I will start building parts of UI components every day, leveraging NextJS features for ease, and my already existing knowledge of TypeScript which I use every day in work – this is something I am comfortable with and hopefully will be in my favour over the coming weeks. I have also began researching Auth Providers with more simple setup – in an attempt to outsource the burden of comprehensive auth to people who know what they are doing. This will cut down on actual development time and I am hoping it can shave down the already missed time, and I suppose if I have time at the end I can simply complete it from scratch.</p>	
Student Signature	<i>Lee Campbell</i>

Supervision & Reflection Template

Student Name	Lee Campbell
Student Number	X20115075
Course	Computing – 4 th Year
Supervisor	Shivani Jaswal

Month: April

<p>What?</p> <p>Reflect on what has happened in your project this month?</p> <p>Over the last month, I have even more progress which I am happy with. I made a diagram which explains the authorisation flow for getting the bank account data and have tested it locally using tools like Postman and the CLI with curl commands. I have updated my projects processes and have setup a new GitHub. I am using Linear, workflow management tool for documenting my progress and creating tickets. I also learned about Server Actions and API Routes in NextJS, and decided to implement simple API Routes for fetching and making the initial GoCardless(Banking) requests. I rejigged the projects structure to make it more in line with what I had seen in other NextJS Project GitHub, and have chosen an Auth Provider in Kinde, testing both the login/signup functionality, bonus is it allows logging in/logging out with Google which is a nice touch, for both me and the potential “users”. I have also built a dashboard, with a number of subheadings , one of these headings is chat with pennywise, comprising of a API Route that creates a simple chat with AI functionality, and streams into back into the UI. I also managed to create API routes for passing the secrets to the Banking API, and getting back a list of banking institutions and logos, which I can show in a UI component.</p>
<p>So What?</p>

Consider what that meant for your project progress. What were your successes? What challenges still remain?

My project is coming along really nicely, I can already see that this is achievable, having built the majority of the UI and understanding more about how NextJS works under the hood, this allowed me to actually deliver, I felt as if this last month was make or break, and I managed to make.

Challenges that remain include, the banking flow, although I made the diagram, and can show the banking institutions, there is a number of requests that need to be made behind the scene, currently trying to figure out how best to do this, possible loading screen or something similar, that has proved difficult. Also, since I built so quickly, I have not setup the cypress for testing UI components. This will be added but have not had the time to make something appropriate for the midpoint presentation. One more thing is creating User Sessions, from researching, it's better to save accessTokens and secrets into a user session that is saved in a database, so this is something that needs more research.

Now What?

What can you do to address outstanding challenges?

Research more in depth on user sessions, complete the banking flow and think about loading later or what to display to the user later, create more refined tickets and again, time management is always key.

Student Signature

Lee Campbell

Please fill in the following sections, if you think your idea is innovative:

1. Title of Invention

PennyWise, Financial Insights

2. Inventors

Name	School/Research Institute	Affiliation with Institute (i.e. department, student, staff, visitor)	Address, contact phone no., e-mail	% Contribution to the Invention

Lee	NCI	Student	X20115075@student.ncir l.ie	100%
-----	-----	---------	--------------------------------	------

3. Contribution to the Invention

Each contributor/potential inventor should write a paragraph relating to his/her contribution and include a signature and date at the end of the paragraph.

My contribution is I have built this application from the ground up, I originally had the idea around one year ago, and decided to create this application for my final year computing project, I have always been interested in building my own things, and this was my first idea for something to build by myself that I thought could be completed.

Lee Campbell

4. Description of Invention

(Please highlight the novelty/patentable aspect. Attach extra sheets if necessary including diagrams where appropriate). What is novel, the 'inventive step'? For more information on patents, please look at <http://www.patentsoffice.ie/en/patents.aspx>

Integration of AI Models – for insights rather than advice. Improvements upon this POC could lead to automated budgeting, budgets alerts, re-structuring, and being able to converse with “wallet”, something that knows your spending habits and available funds.

Seamless Banking Connectivity. Join any bank available in Ireland. One centralized place for all your accounts, and the different information within.

I/we acknowledge that I/we have read, understood and agree with this form and the Institute's *Intellectual Property and Procedures* and that all the information provided in this disclosure is complete and correct.

I/we shall take all reasonable precautions to protect the integrity and confidentiality of the IP in question.

Inventor: *Lee Campbell*

Date: 5th August, 2024.