code sprint MT NATIONAL CODING COMPETITION

Competition Booklet

Open Competition 2025

POWERED BY medirect

GOVERNMENT OF MALTA
MINISTRY FOR EDUCATION,
SPORT, YOUTH, RESEARCH
AND INNOVATION

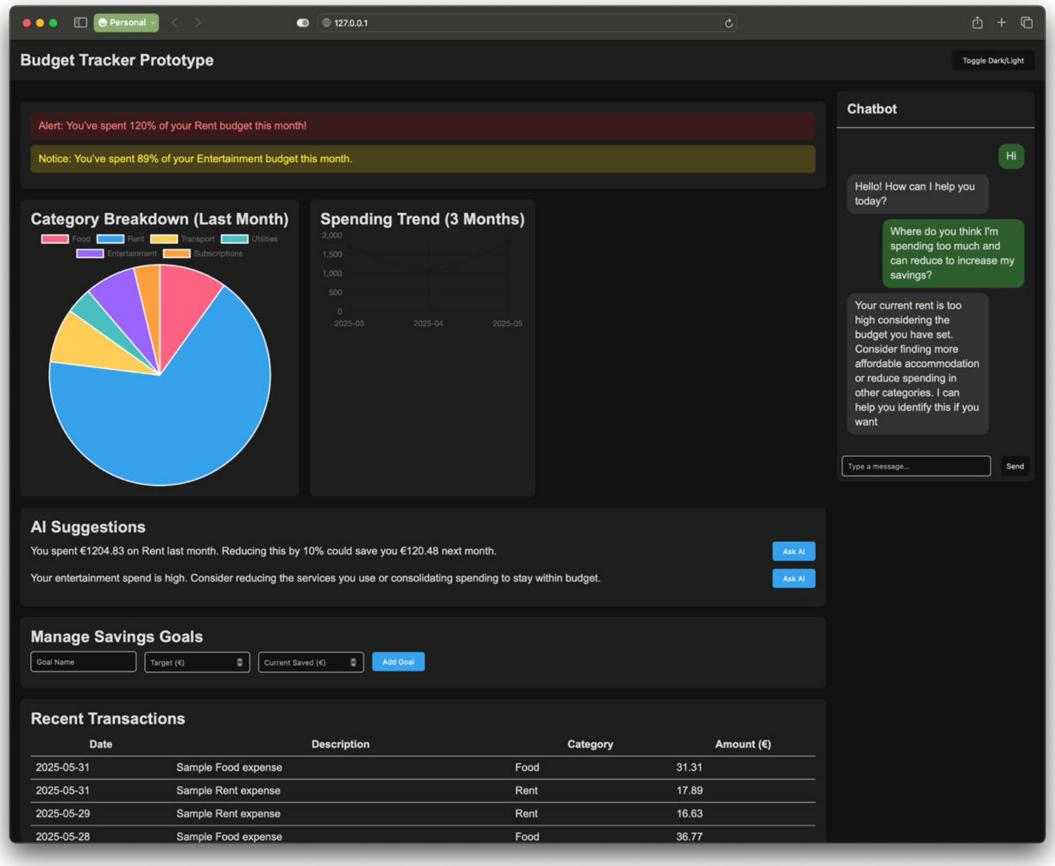
×

icecampus

SmartSave – Personalised Savings Assistant

Hello!

Welcome to the code.sprint^{mt} 2025 competition. Today you'll be flexing your coding, infrastructure and UI/UX design muscles to create a practical lab learning environment.



Alert: You've spent 120% of your Rent budget this month!

Notice: You've spent 89% of your Entertainment budget this month.

Hi

Hello! How can I help you today?

Where do you think I'm spending too much and can reduce to increase my savings?

Your current rent is too high considering the budget you have set. Consider finding more affordable accommodation or reduce spending in other categories. I can help you identify this if you want

AI Suggestions

You spent €1204.83 on Rent last month. Reducing this by 10% could save you €120.48 next month.

Your entertainment spend is high. Consider reducing the services you use or consolidating spending to stay within budget.

Manage Savings Goals

Recent Transactions

Date	Description	Category	Amount (€)
2025-05-31	Sample Food expense	Food	31.31
2025-05-31	Sample Rent expense	Rent	17.89
2025-05-29	Sample Rent expense	Rent	16.63
2025-05-28	Sample Food expense	Food	36.77

1. Design Brief

One of the biggest parts of 'adulting' is managing your money. In today's world, inflation and a barrage of subscription services make it hard to determine exactly where your salary ends up each month. The goal of SmartSave is to create a dashboard for the average person to be able to gain insights into their expenses each month.

SmartSave should help users to develop good spending habits by offering personalized advice, based either on rules, AI or, ideally, a combination of both.

Your challenge is to create a digital assistant that provides tailored savings insights using mock financial data and smart logic.

Since you only have a day, the app you will be building will be a prototype / proof-of-concept. Below is a table of functionality prioritized using the MoSCoW method.

Must Have

Functionality		Notes
M1 Dashboard Interface		
	M1.1	Present a clear view of a user's past three months of expenses. You have been provided with mock data for this.
	M1.2	Categorise spending into buckets (e.g. food, rent, transport, subscriptions).
	M1.3	Visualising spending (including trends) using charts/graphs.
M2 Savings Goal Engine		
	M2.1	Allow users to set one or more savings goals (e.g. vacation, emergency funds).
	M2.2	Automatically suggest how much a user could save per month, based on their spending patterns.
	M2.3	Provide a recommendation engine that suggest areas to cut back on (e.g. "Reducing your delivery food spend by 15% could save you €100/month").

Should Have

Functionality		Notes
S1 Intelligent Alerts		
	S1.1	A basic rule engine to simulate alerts (e.g. "You've spent 80% of your good budget this month!").
S2 Natural-Language Assistant		
	S2.1	Use a chatbot (or any other natural-language interface) for user interaction, allowing the user to ask questions about their spending patterns and data.

Could Have

Functionality		Notes
C1 Browser Extension		
	C1.1	A browser extension (for Chrome-based, Gecko-based or WebKit-based browsers). This extension should provide alerts when new transactions are recorded. Notifications can be triggered manually for the purposes of this prototype.
C2 WhatsApp Integration		
	C2.1	Build a WhatsApp integration which sends push notifications to the user's WhatsApp account. Notifications can be triggered manually for the purposes of this prototype.

Won't Have

We recommend you do not attempt the following, due to time constraints and complexity:

1. Actual integration with an OpenBanking API (even a mock one). This often involve complex oAuth2 flows which are overkill for a prototype.
2. Publication of a browser extension to a store. This requires verification and, depending on browser, payment – which is not required for this competition.

2. Technical Guidelines

The method you choose to implement the app is up to you. However, the following technical guidelines are intended to help ensure you stay on the right track.

2.1 Sample Data

Sample transaction data has been created for you to source. You can find it below:

https://codesprint25.s3.eu-central-1.amazonaws.com/codesprint_open_2025_sample_data.csv

2.2 Reference Implementation

The judging panel has created a reference implementation of this app in one working day. This is to ensure that the task given is possible within the timeframe allocated. A video of this app in operation is available below. We highly recommend that you watch this video carefully, to get an idea of the functionality and level of polish the judging panel is expecting.

https://icepublicvids.s3.eu-south-1.amazonaws.com/SmartSave.mp4

2.3 Platform

Your app can run on any platform you choose (Windows, macOS, Linux, iOS, Android, Web). We suggest creating a web application to reduce the barrier of entry for users (as well as simplicity!) however this is ultimately up to you.

2.4 Development Environment

You are free to use **ANY** programming language you wish to create your solution. However, do remember that the solution must run on the judge's computers, and that you must provide both a binary/executable solution, as well as source code.

2.5 Tips

Here are a few tips and resources to consider.

- Due to time constraints, you may want to seriously consider building a simple web application for this proof-of-concept, rather than a mobile or desktop application.
- Use of AI assistance in coding is not only accepted, but encouraged. However, as with any AI, do check the results of your prompts for functionality, correctness and security. You will be required to explain random sections of your code during your project VIVA.

2.6 Resources

Rendering Charts & Graphs

- On the web: https://www.chartjs.org/docs/latest/
- Android:
  - https://github.com/ehsannarmani/ComposeCharts
  - https://github.com/PhilJay/MPAndroidChart
- iOS: https://developer.apple.com/documentation/charts

Interacting with a REST API

- https://restfulapi.net
- https://www.youtube.com/watch?v=Q-BpqyOT3a8

LLM APIs

- OpenAI: https://platform.openai.com/docs/overview
- Microsoft Copilot: https://learn.microsoft.com/en-us/microsoft-365-copilot/extensibility/copilot-apis-overview
- Anthropic Claude: https://www.anthropic.com/api
- Google Gemini: https://ai.google.dev

Open Banking API

Note: for the purposes of this task, you do not need to learn the Open Banking API! You will however need to mock an end point for basic, read-only use. Implementing an oAuth2 authentication flow is similarly not required.

- Mock bank (free to use): https://www.mockbank.io.
  Also check out their documentation at:
  https://jrholding.atlassian.net/wiki/spaces/MPD/pages/685670401/Introduction+to+MockBank
- Mocking an API:
  - https://designer.mocky.io
  - https://github.com/typicode/json-server

Browser Extension Development

- Chrome-based: https://developer.chrome.com/docs/extensions/
- Firefox (Gecko)-based: https://extensionworkshop.com
- Safari (WebKit)-based: https://developer.apple.com/documentation/safariservices/safari-web-extensions

Creating a WhatsApp Integration

- https://developers.facebook.com/docs/whatsapp/
- https://www.youtube.com/watch?v=4cvQxqFZTIQ
- https://medium.com/%40ammarbinshakir557/whatsapp-api-integration-with-node-js-f915cad3cc3b

2.7 Name

SmartSave is a sample name – you are free to call your app whatever you want ☺

3. Judgement Criteria

Your submission will be given a maximum of 210 points. The criteria by which points are awarded are detailed below. **Note that you do not need to achieve all the criteria**, however, the more criteria you achieve, the greater your chances of winning! The numbers in **[brackets]** refer to the functionality in the design brief.

Criterion	Notes	Maximum Points
Core Functionality		
[M1.1] Expense Viewing	The user should be able to view their past three months of expenses. This should be presented in a sortable and ideally, searchable, form.	**10**
[M1.2] Spending Categorisation	Provided with the mock data, the application should automatically categorise spending into 'buckets' such as food, rent, transport etc…	**20**
[M1.3] Spend Visualisation	Spending should be visualised. At a bare minimum, a pie chart showing spend categorised by bucket and a graph showing spending trends by month should be provided.	**15**
[M2.1] Savings Goals	Users should be able to set savings goals (e.g. vacation, emergency funds). Users can then deposit funds into these virtual spaces, with the system tracking amount left to target.	**15**
[M2.2] Saving Suggestion	The system should display an automated suggestion showing how much (in total) the user can save per month, based on their savings patterns.	**15**
[M2.3] Recommendation Engine	Provide a recommendation engine that suggests areas to cut back on (e.g. "Reducing your delivery food spend by 15% could save you €100/month).	**15**
UI/UX		
Neat/Aesthetically pleasant user interface	Rather than 'flair', we are looking for a neat, organized and functional UI	**10**
App is easy to use	The user should not need a manual to use the app	**5**
Responsiveness	The app should be usable on different screen sizes.	**10**
Code Quality		
Code is organized into packages/modules/units etc.		**5**
Separation between presentation and logic layers	For example, using a REST API model	**10**
Consistent and correct use of a programming paradigm	Such as OOP, AOP, functional etc.	**5**
Function cohesion	Functions should be kept small, and do one thing, without being too dependent on other functions	**5**
Inline documentation	i.e. comments	**5**
Maintainable code	Ex: use of abstract classes, interfaces, function prototypes etc. Depending on the programming paradigm chosen	**5**
Additional Functionality/Features		
[S1.1] Alerts	A basic rule engine to simulate alerts (e.g. You've spent 80% of your food budget this month).	**10**
[S2.1] Natural Language Assistant	Using a natural-language interface to allow user interaction with their data.	**20**
[C1.1] Browser Extension	A browser extension to provide spending alerts.	**15**
[C2.1] WhatsApp Integration	WhatsApp integration to provide spending alerts.	**15**
Additional features	*Additional features not present in these requirements will be graded up to a maximum of 15 marks.*	

Submission Criteria

At the end of the time allocated to this competition, you must submit your code to the judging panel. The code, including all assets and other resources, must be submitted as a folder or compressed archive.

You will also be required to demonstrate your application running.

NOTES

code sprint MT NATIONAL CODING COMPETITION

ORGANISED BY

GOVERNMENT OF MALTA
MINISTRY FOR EDUCATION,
SPORT, YOUTH, RESEARCH
AND INNOVATION

× icecampus

POWERED BY TOP BRANDS

MAIN TITLE SPONSORS

mdia MALTA DIGITAL INNOVATION AUTHORITY

medirect

COMMUNITY & EXPERIENCE SPONSORS

melita

Bolt Business

KNIGHTS COLLEGE

pwc

J'Oli

SUPPORTING SPONSORS

Tech.mt MALTA - LEADING THROUGH INNOVATION

XJENZA MALTA

CISCO Networking Academy

9H

logicom

NGU TECHNOLOGIES

MEDIA PARTNERS

Lovin Malta

TIMES OF MALTA

meet INC.