codesprint

NATIONAL CODING COMPETITION

Competition Challenge

Open Category 2026

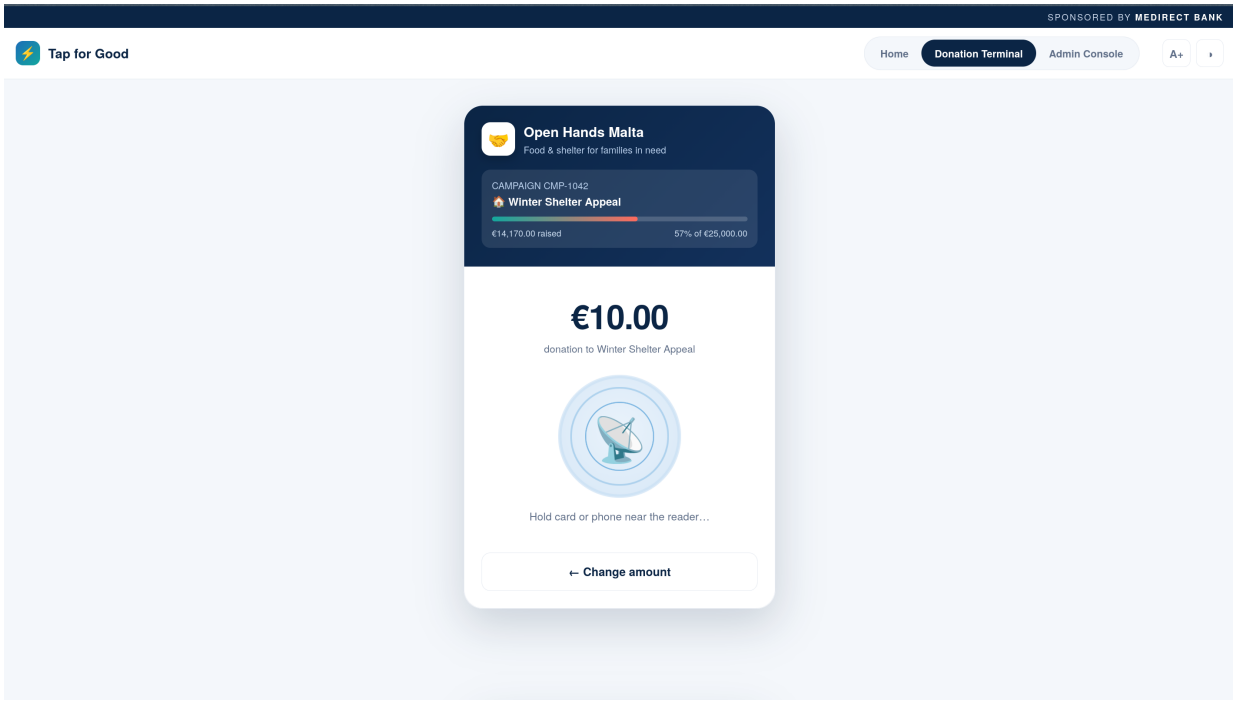
POWERED BY me direct

GOVERNMENT OF MALTA
MINISTRY FOR EDUCATION AND SPORTS
DIRECTORATE FOR STEM AND VET PROGRAMMES

× icecampus

Tap For Good – Contactless Giving for Charities

Hello! Welcome to the CodeSprintMT 2026 competition. Today you'll be flexing your coding, infrastructure and UI/UX design muscles to create an app to help worthy causes.



1. Design Brief

Cash is disappearing. Coffee runs, transit, and groceries are all contactless - but the church collection plate, the street fundraiser, and the school bake sale are still cash-first. Charities lose out on spontaneous giving every day because they can’t take a card. Your challenge is to close that gap.

2. Requirements

Since you only have a day, the app you will be building will be a prototype / proof-of-concept. Below is a prioritised requirements list. **Note:** Failing to meet a core requirement does not result in a failure. The prioritisations below are suggested.

Core Requirements

Functionality		Notes
M1 User Front-end ***(Donation App)***		
	M1.1	Campaign and charity branding (logo, cause, suggested amounts)
	M1.2	Preset amounts (€5, €10, €25) plus custom amount entry
	M1.3	Tap-to-donate flow (simulated) and card details entry against the Mastercard API sandbox
	M1.4	Confirmation screen and optional email or SMS receipt
	M1.5	Multi-currency support (EUR primary)
	M1.6	Graceful failure handling (declined, cancelled, offline)
	M1.7	Accessibility (large-text mode, high contrast)
M2 Admin Front-end ***(Admin tool)***		
	M2.1	Authentication and role-based access (Charity Admin, Volunteer, Auditor)
	M2.2	Create and manage campaigns (name, goal, suggested amounts, start and end dates)
	M2.3	Live dashboard: total raised, donor count, average donation, progress vs. goal
	M2.4	Transaction ledger filterable by date, campaign, status, and amount
	M2.5	CSV and PDF export for accounting
	M2.6	Reconciliation view tying sandbox transaction IDs to internal donation records (only for card details payments)
	M2.7	Audit log of admin actions

Important Requirements

Functionality		Notes
S1 Scenario Usability		
	S1.1	Recurring donations, or “round up my next purchase” (only for card details payments)
	S1.2	Tax-deductibility or Gift Aid-style declaration capture
	S1.3	Donor recognition mode (named vs. anonymous)

Optional Extras

Functionality		Notes
C1 Mobile App		
	C1.1	A simple mobile app demonstrating use of the application on a mobile device, as a proof-of-concept. Using a webview (rather than a native app) is acceptable
C2 Internationalisation		
	C2.1	Multilingual UI (EN / MT / FR / NL - (reflecting MeDirect's geographies)
C3 Real Time		
	C3.1	Admin dashboard updated in real time (for example, via websocket)

Won't Have

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

1. Actual integration with a payment processor.
2. Publication of the mobile app to an app store.

Note: The Mastercard Sandbox API does not allow 'tap to pay' for donation payments. Therefore, you're only required to simulate this functionality. However, card details payments (i.e. Where the user enters their card number, expiry date and CVV) must validate against the sandbox API.

3. Demonstration Video

Please read this section **VERY CAREFULLY**.

You must create a demonstration video as part of your submission. This video must showcase EACH and EVERY feature you have implemented in your solution. In your video, make sure you go over all of the functionality of both the donation app and admin tool. Ideally, the video should be narrated by yourself.

FAILURE TO SUBMIT A VIDEO WILL RESULT IN IMMEDIATE DISQUALIFICATION

ANY FEATURES IMPLEMENTED IN YOUR SOLUTION WHICH ARE NOT DEMONSTRATED IN THE VIDEO WILL NOT BE GRADED. THIS INCLUDES OPTIONAL (NON-CORE) OR ADDITIONAL FEATURES.

4. 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.

4.1 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/codesprint2026.mp4

4.2 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.

4.3 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.

4.5 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 the core and important parts of your solution (M and S), 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.
- The donate APIs integration requires manual submission of the card details rather than a tap to pay. For this reason you are only expected to simulate an endpoint call, not actually make it.

4.6 Resources

For your reference, here are some resources you may want to consult. Note, you will not need all of them!

The MasterCard Sandbox

- Creating a Mastercard Sandbox Account
  https://developer.mastercard.com/donations/documentation/quick-start-guide/#generate-credentials-and-create-a-sandbox-project
- Mastercard Donate (purpose-built for this use case)
  https://developer.mastercard.com/product/mastercard-donate/
- Donate API - documentation (donor/card enrolment, donation processing)
  https://developer.mastercard.com/donations/documentation
- Donate API - API reference
  https://developer.mastercard.com/donations/documentation/api-reference/
- Donate API - API basics
  https://developer.mastercard.com/donations/documentation/api-basics/
- Mastercard Postman workspace (call the APIs before writing code)
  https://www.postman.com/mastercard/workspace/mastercard-developers/overview

Payments & Contactless Fundamentals

- EMVCo (the standards body behind chip & contactless "tap")
  https://www.emvco.com/
- Stripe Docs: Tap to Pay (clearest plain-English explainer of phone-as-reader)
  https://docs.stripe.com/terminal/payments/setup-reader/tap-to-pay
- Google Pay: Tap to Pay overview
  https://developers.google.com/pay/issuers/tap-to-pay

Mobile App & NFC (The Donation Terminal)

- Flutter docs
  https://docs.flutter.dev/
- React Native docs
  https://reactnative.dev/docs/getting-started
- Android NFC basics
  https://developer.android.com/develop/connectivity/nfc

Admin Tool / Dashboard

- React
  https://react.dev/learn
- Next.js docs
  https://nextjs.org/docs
- Vue
  https://vuejs.org/guide/introduction.html
- Tailwind CSS
  https://tailwindcss.com/docs
- Recharts (live KPI/progress charts)
  https://recharts.org/

- Chart.js
  https://www.chartjs.org/docs/latest/
- TanStack Table (sortable/filterable transaction ledgers)
  https://tanstack.com/table/latest

Backend & APIs

- Node.js docs
  https://nodejs.org/en/docs
- Express
  https://expressjs.com/
- FastAPI (Python)
  https://fastapi.tiangolo.com/
- ASP.NET Core
  https://learn.microsoft.com/en-us/aspnet/core/
- REST API design guidelines (Microsoft)
  https://github.com/microsoft/api-guidelines

Auth & Role-Based Access

- Auth0 docs
  https://auth0.com/docs
- Clerk docs
  https://clerk.com/docs
- jwt.io - Introduction to JWTs
  https://jwt.io/introduction
- Auth0: Role-Based Access Control (RBAC)
  https://auth0.com/docs/manage-users/access-control/rbac

Database

- PostgreSQL docs
  https://www.postgresql.org/docs/
- Supabase
  https://supabase.com/docs
- Prisma ORM
  https://www.prisma.io/docs

Webhooks & Real-Time Dashboards

- webhooks.fyi (vendor-neutral best practices: signing, retries, idempotency)
  https://webhooks.fyi/
- ngrok docs (expose localhost to receive sandbox webhooks during dev)
  https://ngrok.com/docs
- Socket.IO (push live donation updates to the dashboard)
  https://socket.io/docs/v4/

Accessibility, Internationalisation & Multi-Currency

- WCAG at a glance (W3C/WAI) - large-text & high-contrast requirements
  https://www.w3.org/WAI/standards-guidelines/wcag/
- WebAIM Contrast Checker (verify high-contrast mode in seconds)
  https://webaim.org/resources/contrastchecker/

- i18next (multilingual UI: EN / MT / FR / NL)
  https://www.i18next.com/
- FormatJS / react-intl
  https://formatjs.io/docs/getting-started/installation/
- MDN: Intl.NumberFormat (correct multi-currency formatting)
  https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Intl/NumberFormat
- ISO 4217 currency codes
  https://www.iso.org/iso-4217-currency-codes.html

Diagrams, Readme, Exports

- C4 model (architecture diagrams)
  https://c4model.com/
- Mermaid (diagrams as code)
  https://mermaid.js.org/
- Excalidraw
  https://excalidraw.com/
- draw.io
  https://app.diagrams.net/
- makeareadme.com
  https://www.makeareadme.com/
- Papa Parse (CSV export)
  https://www.papaparse.com/
- pdfmake (PDF export)
  http://pdfmake.org/
- jsPDF (PDF export)
  https://github.com/parallax/jsPDF

5. Judgement Criteria

Your submission will be given a maximum of 345 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 references in **[brackets]** refer to the functionality in the design brief (see section 2).

Criterion	Notes	Maximum Points
Core Functionality		
[M1.1] Branding	Organisation logo, colour scheme and suggested amounts displayed in donation app	**10**
[M1.2] Preset Amounts	Ability for donor to select from pre-set amounts or specify a custom amount	**10**
[M1.3] Donation Flow	Donation flow once an amount has been selected, including integration with MasterCard sandbox for card details payments.	**20**
[M1.4] Confirmation Screen & Receipt	Display of confirmation screen and ability for donor to receive confirmation/receipt via email or SMS	**20**
[M1.5] Multi-Currency Support	Ability to support at least three currencies, with EUR as the primary	**15**
[M1.6] Graceful Error Handling	Validation and error handling for incorrect input or exceptions	**15**
[M1.7] Accessibility	Availability of text size controls and colour scheme swap for high-contrast	**10**
[M2.1] Role-Based Access Control	Logging into the admin tool should allow for increasing functionality based on role (Auditor < Volunteer < Admin)	**20**
[M2.2] Create & Manage Campaigns	Ability to create, edit and delete campaigns with fields for name, goal, suggested amounts, start and end dates	**20**
[M2.3] Live Dashboard	Total raised, donor count, average donation, progress towards goal	**20**
[M2.4] Transaction Ledger	List of all transactions carried out in app. Filterable and sortable	**15**
[M2.5] CSV & PDF Export	Transaction ledger exportable as CSV or PDF	**10**
[M2.6] Reconciliation View	Ability to match transaction records from Mastercard sandbox to donation records from your app	**10**
[M2.7] Audit Log	Log of every admin action	**10**
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 both desktop and mobile screens	**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] Recurring Donations	UI prototype for when a user chooses "round up my next purchase" or a recurring donation. Only for card details payments.	**10**
[S1.2] Tax Deductibility / Gift-Aid	UI prototype for a tax deductibility / gift-aid style form approved by users (does not need to be legally accurate)	**10**
[C1.3] Donor Recognition	Ability for donors to choose to have their name displayed as a donor on the donation app front page. Choosing this option must display their name in the front page (i.e. Not just a mock form)	**20**
[C1.1] Mobile App	Prototype of mobile app	**15**
[C2.1] Multilingual UI	Support for translations of the UI, with at least one partial translation to another language	**15**
[C3.1] Real-time dashboard	Admin dashboard updated in real time on donation	**10**
Additional features	*Used as a tie-break. See below.*	

Additional Features

Should you wish to add features to your solution which are over-and-above the requirements and suggestions above, you are free to do so. However, please note that **these will not be graded**.

Additional features are only considered in the unlikely event that two submissions receive the exact same grade. In this case, the winner between the two will be decided by a VIVA as well as additional features provided, and is at the sole discretion of the judging panel.

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.

Remember to include your demonstration video in the submission.

6. AI-Assisted Coding

Intro
The aim of the CodeSprintMT competition is to reward developers who can solve problems quickly and creating prototypes and proof-of-concept apps for real-world scenarios. In 2026, that will inevitably involve the use of AI-assisted development.

Ok, so can I use AI?
Yes. In fact, it is expected that you will.

Do I have to comment sections created by AI for attribution?
No. AI-created code is a result of your prompt and existing work. The AI does not "own" any code it generates, you do. Commenting sections created or enhanced with AI is impractical and results in messy code files.

Will the use of AI be penalised?
No, if you understand what the AI has created for you. During the VIVA of your submission, you may be asked to explain certain parts of your code. If you can fully explain what your code is doing, there is no penalty for using AI, or indeed the need to disclose which parts of your code were written by AI.

Note, however, that you should always verify any code generated by AI (or indeed, code samples you retrieve via traditional means) for correctness and security. If your code is messy or insecure, you will be penalised, regardless of who or what wrote it!

7. Rules and Regulations

1. CodeSprintMT is open to anyone residing within the Maltese Islands who codes - whether as a student, graduate, working professional, educator, hobbyist or self taught coder. Participation is confirmed upon successful registration and formal acceptance by the organisers, at their sole discretion.
2. Participants are free to use any tech stack of their choice, including programming languages, frameworks, libraries, and tools they are most comfortable with.
3. The session will be held in-person at ICE Campus (Zebbug) using a bring-your-own-device model. Internet access and an optional extended monitor will be provided. Participants are responsible for ensuring their device is fully functional, pre-loaded with any required software or tools, and equipped with the necessary cable to connect to the extended monitor.
4. All participants will work individually (this is a solo challenge). Strict security protocols will be in place throughout the competition. Any attempt to plagiarise or receive help from another person will result in immediate disqualification.
5. Participants are expected to maintain a respectful environment during the session. This includes keeping noise to a minimum, avoiding unnecessary conversation, and not distracting others. Silence is to be observed throughout the competition room unless speaking with an organiser or judge.
6. Participants are allowed to use AI tools responsibly as part of their workflow. You are not required to comment or attribute AI-generated sections, but you must understand your code and may be asked to explain it during the viva. Messy, insecure, or unverifiable code will be penalised. For full AI usage guidelines, refer to the AI-Assisted Coding section in this document.
7. During the session, it is mandatory that participants:
    a. maintain healthy habits and take short breaks,
    b. save and back up their work regularly throughout the session,
    c. follow all instructions given by organisers and invigilators,
    d. submit their task, including the completed solution and demonstration video, by 5.00PM.
8. Following the competition, all submissions will be reviewed by the judging panel. The top 10 ranked submissions will be shortlisted for a VIVA, held online a couple of days after the competition. Shortlisted participants will be notified by email and given a scheduled time slot.
9. Failure to submit a demonstration video will result in immediate disqualification. Failure to demonstrate a feature in the demonstration video will mean this feature will be ignored by judges during grading.
10. Additional features implemented in a participant's solution which are over-and-above the requirements mentioned in the task booklet will not be graded, unless there is a perfect tie between two participants. In this case, the tie will be resolved via a combination of participant VIVA and any additional features implemented. This is at the sole discretion of the judging panel.

11. The competition area will be monitored by CCTV throughout the session for security and integrity purposes. Footage is retained only for as long as necessary for these purposes.
12. Photographs and video recordings will be taken during the event for marketing and recap purposes. Participants who do not wish to appear in marketing content may notify the organisers before the competition starts and reasonable steps will be taken to accommodate this.

codesprint

ORGANISED BY

GOVERNMENT OF MALTA
MINISTRY FOR EDUCATION AND SPORTS
DIRECTORATE FOR STEM AND VET PROGRAMMES

× icecampus

POWERED BY TOP BRANDS

MAIN TITLE SPONSORS

mdia MALTA DIGITAL INNOVATION AUTHORITY

me direct

COMMUNITY & EXPERIENCE SPONSORS

pwc

melita

Bolt

KNIGHTS COLLEGE

JOli

XJENZA MALTA

SUPPORTING SPONSORS

mita

edQuanta

computime BUSINESS SYSTEMS ENGINEERING

9H

MEDIA PARTNERS

Lovin Malta

TIMES MALTA

meet INC.