Easy2Wakaf

DOING WAQAF NEVER BEEN THIS EASY

Team Name

“That One Time When Me And Bros Decided To Join Hackathon And We Actually Win”

Institution

University of Malaya

Submission For

UMHackathon 2025

Problem Statement

Enhancing Charity & Donations through Fintech & Technology

Date

21st April 2025

Easy2Waqaf: Introduction

Easy2Wakaf is a comprehensive digital solution designed to modernize and unify the Waqf donation ecosystem in Malaysia. Rooted in the principles of Islamic social finance, Waqf has long served as a powerful tool for community development, education, and welfare. However, traditional Waqf systems often suffer from fragmented management, limited accessibility, and transparency concerns that hinder donor trust and engagement. The youth also lacks awareness about the benefits of waqf as a system of welfare.

Easy2Wakaf addresses these challenges by digitizing and centralizing the donation process, creating a secure, Shariah-compliant platform that connects donors to verified Waqf projects across all Malaysian states, starting in Selangor. The platform enhances transparency, boosts donor confidence, and streamlines the collaboration between state Waqf boards.

Built using Django for the backend and React for the frontend, the system also integrates document verification tools and analytical dashboards to ensure trust and impact at every level. With blockchain-inspired ledger tracking and smart visualizations, donors gain real-time insights into how their contributions are used and the tangible outcomes achieved.

Whether contributing to the construction of mosques, supporting education, or funding healthcare initiatives, Easy2Wakaf empowers users to give with confidence and clarity.

Easy2Waqaf: Tech Stack

Technology	Purpose & Description
Django	A powerful Python-based web framework used to build the backend API, handle authentication, and manage admin functionalities with ease and security.
React	JavaScript library for building dynamic, responsive single-page applications (SPA) for both donors and administrators.
Firebase	A NoSQL database chosen for its scalability and simplicity in handling diverse data types, ideal for managing donation records and project information.
Stripe	Integrated as a secure and reliable payment gateway to process online Waqf donations. Supports multiple currencies and ensures donor trust.
OpenCV	An open-source computer vision library used to scan and verify identity documents and detect logos (e.g., PWS) as part of project legitimacy checks.
Gemini (AI)	Employed for AI-powered insights, document parsing, and potential automation of verification workflows to enhance scalability and intelligence.
Power BI	Interactive dashboard tool for visualizing donation flow, project impact, and financial analytics in real-time, improving transparency and trust.
Canva / Figma	Used for designing high-fidelity UI/UX prototypes, ensuring the platform is user-friendly, accessible, and visually appealing across all devices.

Easy2Waqaf: Installation & Setup

This section outlines how to set up the Easy2Wakaf system locally, including both the backend (Django) and frontend (React) applications.

Backend (Django)

Prerequisites:

- Python 3.8+
- pip
- Firebase
- Virtualenv

Steps:

1. **Clone the repository:**

   ```
   git clone https://github.com/your-username/easy2wakaf.git
   cd easy2wakaf/backend
   ```

2. **Create and activate a virtual environment:a**

   ```
   python -m venv env
   source env/bin/activate   # On Windows: env\Scripts\activate
   ```

3. **Install Python dependencies:**

   ```
   Django==5.2
   django-cors-headers==4.7.0
   djangorestframework==3.16.0
   firebase-admin==6.7.0
   stripe==12.0.0
   python-dotenv==1.1.0
   pillow==11.2.1
   requests==2.32.3
   google-cloud-firestore==2.20.2
   google-cloud-storage==3.1.0
   google-auth==2.39.0
   ```

4. **Apply migrations and run the development server:**

```
python manage.py migrate
python manage.py runserver
```

Frontend (React)

Prerequisites:

- Node.js (v16+)
- npm or yarn

Steps:

1. **Navigate to the frontend directory:**

   ```
   cd ../frontend
   ```

2. **Install Node dependencies:**

   ```
   npm install
   # or
   yarn install
   ```

3. **Set environment variables:**

   Create a .env file in the frontend folder:

   ```
   REACT_APP_API_BASE_URL=http://localhost:8000/api
   ```

4. **Start the development server:**

   ```
   npm start
   # or
   yarn start
   ```

Easy2Waqaf: Code Overview

The frontend of Easy2Wakaf is built using React, a modern JavaScript library for building user interfaces. The application follows a Single Page Application (SPA) structure, offering a smooth, fast, and interactive experience to users. It communicates with the backend via RESTful APIs and includes components optimized for both donors and administrators.

The frontend codebase is structured for clarity, reusability, and scalability.

1. Backend Code Overview

The backend is developed using **Django**, a high-level Python web framework. It follows a modular architecture using Django apps, each targeting a specific business function. This structure promotes separation of concerns, better maintainability, and scalability.

1.1 authentication/

This app manages all user-related operations, including account creation, login, logout, and session management.

views.py	The frontend of **Easy2Wakaf** is built using **React**, a modern JavaScript library for building user interfaces. The application follows a Single Page Application (SPA) structure, offering a smooth, fast, and interactive experience to users. It communicates with the backend via RESTful APIs and includes components optimized for both donors and administrators. The frontend codebase is structured for clarity, reusability, and scalability.
urls.py	Maps HTTP endpoints like /register/, /login/, and /logout/ to the corresponding view functions. Ensures clean and RESTful API design.

1.2 waqf/

Responsible for managing Waqf projects and handling incoming donations from users.

models.py	Contains models for Waqf projects (title, description, state, amount) and donations (amount, donor, project).

views.py	Implements logic for adding new projects, Display available projects to users, and Accept and process donation transactions securely.
urls.py	Routes like /projects/, /create-project/, and /donate/ are defined here. They map to views responsible for listing, creating, and donating to projects.

1.3 ledger/

Ensures transparency and financial accountability by maintaining a record of all transactions.

models.py	Defines the Ledger model, which captures metadata for every donation including donor name, project ID, amount, receipt ID, and donation timestamp.
views.py	Enables uploading external financial records (e.g. from banks), Viewing donation logs and summaries and Validating uploaded donation entries
urls.py	Includes routes like /upload-ledger/, /view-ledger/, and /ledger-summary/. These routes provide access to the ledger system for admins and auditors.

1.4 analyser/

Handles real-time document verification and scanning for identity or authenticity using computer vision.

views.py	Uses OpenCV to detect logos like the Perbadanan Wakaf Selangor (PWS) mark, verify the integrity of scanned donation or approval documents and extract data using image processing for future automation
urls.py	Exposes endpoints such as /scan-document/ and /verify-upload/, making this module useful for compliance and fraud prevention.

1.5 Easy2Wakaf/ (Main Configuration)

This is the root Django project directory that links and configures all sub-apps.

settings.py	Contains global configuration: ● Installed Django apps ● Firebase database settings ● Middleware for security and CORS ● Static file paths, logging, and secret keys
urls.py	Provide entry points for deployment environments: ● `wsgi.py` for traditional web servers like Apache or Gunicorn ● `asgi.py` for asynchronous servers like Daphne or Uvicorn
wsgi.py & asgi.py	Serve as entry points for running the app in different environments (WSGI for traditional servers, ASGI for async servers).

1.6 manage.py

This is Django's CLI tool for administrative tasks. Commonly used commands include:

- python manage.py runserver – Launches the local development server
- python manage.py makemigrations – Generates migration files for model changes
- python manage.py migrate – Applies migrations to the database
- python manage.py createsuperuser – Creates an admin user for dashboard access
- python manage.py shell – Opens an interactive Python shell with Django context

2. Frontend Code Overview

The frontend is built using React, a popular JavaScript framework, and is structured for scalability and modular development.

2.1 public/

This directory contains static assets that are publicly accessible.

index.html	The main HTML template that hosts the React app.
Favicon.ico, manifest.json	Support browser tab icons and Progressive Web App features like "Add to Home Scree

2.2 src/

The heart of the frontend application. This folder contains all the logic, components, and assets for the React interface.

App.js	Acts as the main wrapper for all pages. Handles the routing logic using react-router-dom. It defines which page/component is shown for each URL, and also renders shared components like the navbar and footer.
index.js	The entry point of the application. Renders the <App /> component into the root div defined in index.html. Also includes base configurations like wrapping the app in <BrowserRouter> for routing.
components/	Reusable UI components used throughout the app. Promotes DRY (Don't Repeat Yourself) principles and modular design. ● Navbar.js – Handles navigation links, conditionally renders based on login state ● Footer.js – Contains copyright ● Card.js – Displays Waqf project summaries in a uniform, clickable card format ● DonationForm.js – Form for donors to input payment and personal details
pages/	Full-screen views or routes of the application. Each file here

	corresponds to a complete page the user can visit. ● Home.js – Landing page showcasing featured Waqf projects and vision statement ● Register.js & Login.js – Authentication pages for new/existing users ● Projects.js – Lists all available Waqf projects from the backend ● ProjectDetails.js – Shows individual project info, donor form, and past donations ● Dashboard.js – For admin users: shows analytics and management controls
services/	Handles API calls to the Django backend. This abstraction separates the HTTP logic from the UI. ● authService.js – Handles user login, registration, and token storage ● projectService.js – Manages fetching project lists, individual details, and donation submissions ● ledgerService.js – Sends ledger uploads and retrieves donation records for admin viewing ● scannerService.js – Interacts with the document scanner endpoint for OpenCV-based verifications Each service file uses libraries like axios to make HTTP requests and includes token handling for protected routes.
styles/	Contains custom CSS or SCSS files used for styling the app. It can include theme variables, responsive utilities, and component-specific styles.

2.3 package.json

Defines project metadata, dependencies, and scripts. Key uses include:

- Managing libraries like axios, react-router-dom, bootstrap, or tailwindcss
- Useful scripts:
    - npm start – Runs the app in development mode
    - npm run build – Builds a production-optimized version of the app
    - npm test – (If configured) Runs unit tests

2.4 README.md

Contains documentation auto-generated by Create React App (CRA). You can enhance it by adding:

- Setup instructions (how to run locally)
- API endpoints used
- Screenshots or GIFs of the app
- Contribution guidelines