Trackify Documentation

This document contains a ready-to-use **documentation folder** for your Trackify project. It includes setup instructions, architecture overview, API endpoints, developer notes, testing steps, and a hackathon pitch. You can paste each section into separate files under a `docs/` folder in your repo.

Table of contents

README.md

````
# Trackify — Receipt OCR Expense Tracker

Trackify is a Django web application that helps users track income and expenses.
Users can add incomes manually, upload receipts or screenshots (offline/online
payments), and Trackify will use OCR to extract amount, date, description and
classify the expense into categories. The app also provides simple reports and
dashboards.

## Key features
- Upload receipt images (JPEG/PNG) — OCR-driven extraction
- Upload payment screenshots (Google Pay/UPI) — parsed similarly
- Manual expense entry
- Income management
- Category assignment (rule-based + extensible ML)
- Basic reports (category breakdown, monthly totals)

## Quick start
1. Clone the repo
   ```bash
   git clone <your-repo-url>
   cd Trackify/budgetlens
````
````

```
   ```
2. Create a virtualenv and install dependencies
   ```bash
   python -m venv .venv
   .venv\Scripts\activate    # Windows
   source .venv/bin/activate   # macOS / Linux
   pip install -r requirements.txt
   ```
3. Install Tesseract OCR (required)
   - Windows: install Tesseract and note the install path (e.g. `C:\Program
Files\Tesseract-OCR\tesseract.exe`)
   - Ubuntu/Debian: `sudo apt-get install tesseract-ocr`

4. Update settings (see `INSTALL.md`) and run migrations
   ```bash
   python manage.py migrate
   python manage.py createsuperuser
   python manage.py runserver
   ```

5. Visit `http://127.0.0.1:8000/` and use the upload receipt page (core app).

Repo layout
- `budgetlens/` — Django project
  - `core/` — main app (models, views, services)
  - `accounts/` — user auth (if present)
  - `media/` — uploaded receipts
  - `requirements.txt` — Python dependencies
```

---

# INSTALL.md

```
# Installation and Local Setup

## 1. Python & virtualenv
- Python 3.11+ recommended (your repo shows Python 3.13 in venv)
- Create virtual environment and install requirements

## 2. Install dependencies
```

pip install -r requirements.txt

```
## 3. Install Tesseract OCR
- Windows: download installer and install. Note the path (default: `C:\Program
```
```

```
Files\Tesseract-OCR\tesseract.exe`).
- Linux: `sudo apt-get install tesseract-ocr`

## 4. Configure Django settings
Edit `budgetlens/budgetlens/settings.py` and ensure these values are set:

```py
MEDIA_URL = '/media/'
MEDIA_ROOT = BASE_DIR / 'media'

# Ensure APP_DIRS=True in TEMPLATES
TEMPLATES = [
    {
        'BACKEND': 'django.template.backends.django.DjangoTemplates',
        'DIRS': [BASE_DIR / 'templates'],
        'APP_DIRS': True,
        ...
    }
]
```

If you are on Windows and installed Tesseract, set its path in the OCR service (see `core/services/receipt_parser.py`):

```
import pytesseract
pytesseract.pytesseract.tesseract_cmd = r"C:\Program Files\Tesseract-OCR\tesseract.exe"
```

5. Run migrations and start server

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

```
---

# ARCHITECTURE.md
```

Architecture Overview

Trackify is a monolithic Django application with the following components:

- **Models** (`core/models.py`) — `Income`, `Expense`, `UserProfile`.
```

- **Views/Forms** (`core/views.py`, `core/forms.py`) — upload receipt view, dashboard, manual entry.
- **OCR service** (`core/services/receipt_parser.py`) — runs Tesseract, parses text, extracts amount/date/category/description and returns a dictionary.
- **Templates** (`core/templates/core/*.html`) — upload form, success page, dashboard.
- **Static & Media** — receipts saved under `media/receipts/`.

Data flow (Upload Receipt)

1. User uploads file via form → POST to `/core/upload-receipt/`.
2. View saves file (via model `Expense.receipt_image` or `FileSystemStorage`).
3. OCR service opens saved file, performs preprocessing, runs `pytesseract`.
4. `parse_receipt_image()` extracts amount, date, description, category.
5. `create_expense_from_receipt()` creates an `Expense` entry.
6. Dashboard updates and displays extracted values + receipt preview.

```
---

# USAGE.md
```

How to use Trackify (User Guide)

1. Register / Login using Django auth.
2. Go to "Add Income" to add income entries.
3. To add expense by receipt:
4. Go to Upload Receipt page.
5. Choose image file (receipt or payment screenshot).
6. Click Upload.
7. The app will show parsed fields — review and edit if necessary.
8. To add expense manually, use the Manual Entry form.
9. View Dashboard to see category-wise breakdown and recent transactions.

```
---

# API.md
```

Endpoints / Routes

These are the key web routes (rendered pages). For DRF APIs, add endpoints under `/api/`.

- `GET /core/upload-receipt/` — show upload form
- `POST /core/upload-receipt/` — upload image, create Expense
- `GET /core/dashboard/` — user dashboard

- `GET /admin/` — Django admin

Expected POST payload for upload

- `receipt_image` — multipart file

Response: the view renders `receipt_success.html` with `expense` in context.

```
---

# OCR_DEBUG.md
```

Tesseract & OCR Troubleshooting

If you see `Tesseract not available`, do the following:

1. Ensure Tesseract is installed.
2. On Windows add the tesseract executable path to `receipt_parser.py` (or to PATH):

```
pytesseract.pytesseract.tesseract_cmd = r"C:\Program Files\Tesseract-OCR\tesseract.exe"
```

1. Improve OCR accuracy:
2. Convert to grayscale
3. Apply median filter
4. Boost contrast
5. Optionally apply binary thresholding
6. Debugging: print OCR text in terminal

```
print('OCR OUTPUT:\n', text)
```

1. Regex tuning
2. Accept integers and decimals for amount: `r'(\d+[.,]?\d{0,2})'`
3. Accept multiple date formats: `dd/mm/yyyy`, `yyyy-mm-dd`, `dd Mon yyyy` (use additional regex)

```
---

# CONTRIBUTING.md
```

Contributing

- Fork the repo
- Create a feature branch
- Open a PR with clear description

Please include tests for new features where appropriate. Keep the code style consistent with the project.

```
---

# HACKATHON_PITCH.md
```

Trackify — 2-minute pitch

Problem: People lose receipts and spend time categorizing expenses manually.

Solution: Trackify is a privacy-first Django app that digitizes receipts and screenshots using OCR, auto-classifies expenses, and shows clear reports.

Demo flow (30s): 1. Upload receipt image — OCR extracts amount, date, merchant. 2. App auto-assigns category & saves expense. 3. Dashboard shows spending by category and trends.

Unique points: - Works offline (open-source Tesseract), no cloud dependency. - Handles both physical receipts and payment screenshots. - Simple to extend to ML-based categorizer.

Tech: Django, pytesseract, Pillow. DB: SQLite/Postgres.

```
---

# CHANGES.md (Suggested roadmap)
```

Roadmap & Improvements

Short term: - Add API endpoints (DRF) for React frontend - Add unit tests for OCR parsing - Add UI to edit parsed fields before saving

Medium term: - Train simple classifier (Naive Bayes) to improve categories - Add CSV/Excel export - Add user settings for currency and monthly budgets

Long term: - Mobile apps (React Native) - Bank/SMS integration for auto import

```

---

## How I generated this

I scanned your repo uploaded as `Trackify.zip` and produced this documentation package. To add the docs to your repo, create a `docs/` folder and create each file using the corresponding sections above.

If you want, I can now: - Create the `docs/` folder and add these files inside the unzipped repo and produce a zip for download. - Or create a single `docs.md` file inside the repo.

Which do you prefer?
```