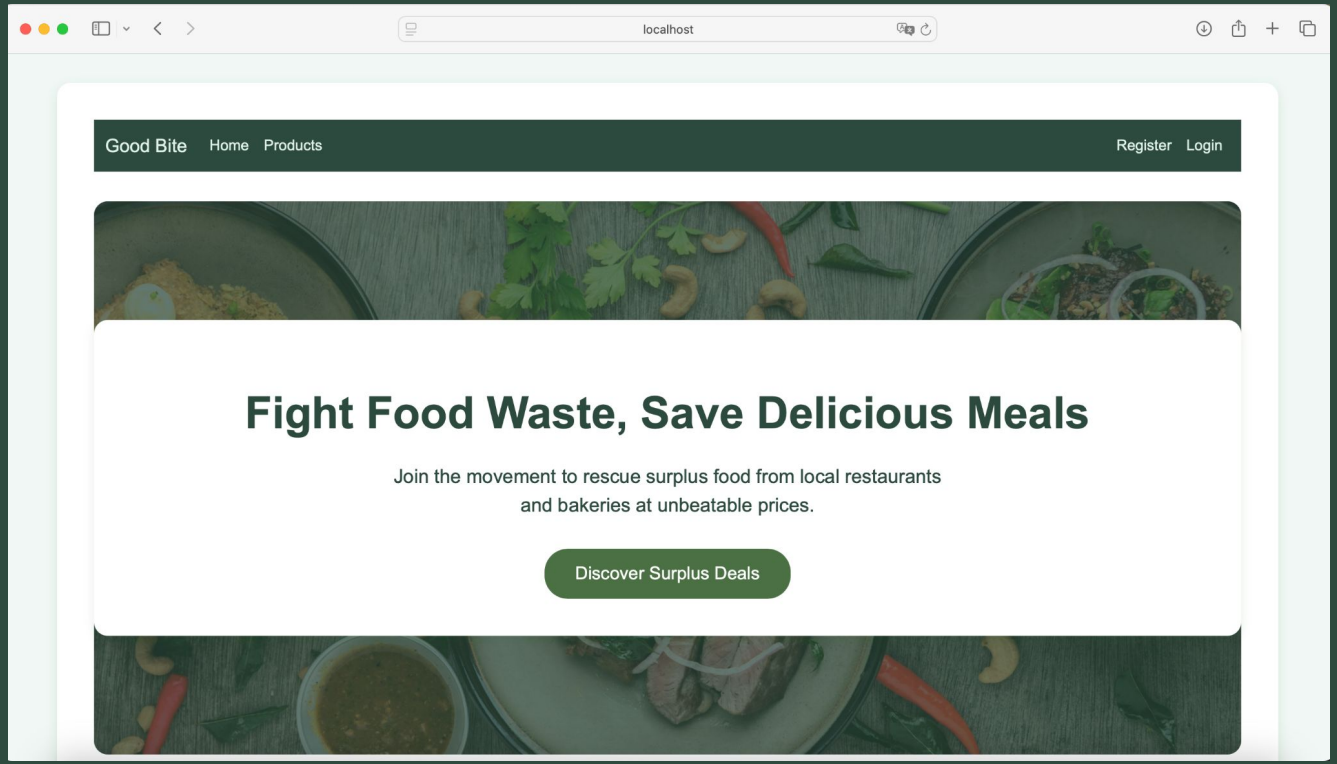
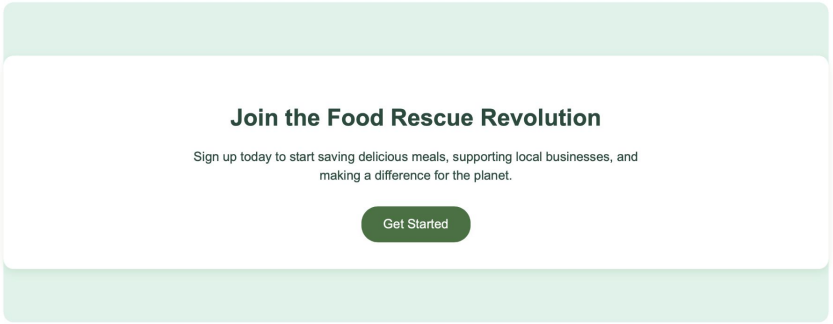
UNIVERSITAT POLITÈCNICA DE CATALUNYA BARCELONATECH

Good Bite App



Anna Melkumyan Canosa · Cecília Maria Rodrigues Correia · Irene Cerván Barriga · Johanna Nuñez Diaz · Oriol Ramos Puig

0. Table of Contents



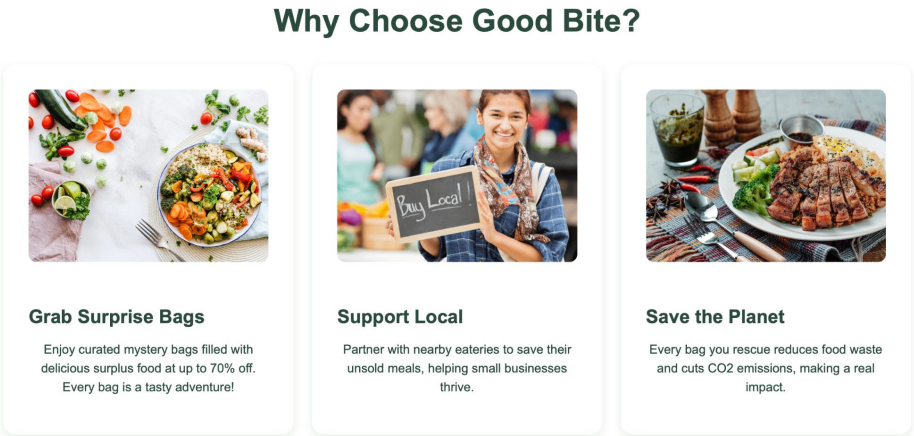
Home page.

Our project repository
GoodBite - Surplus Food Marketplace. 2025.
https://github.com/sissamrcorreia/AS-project

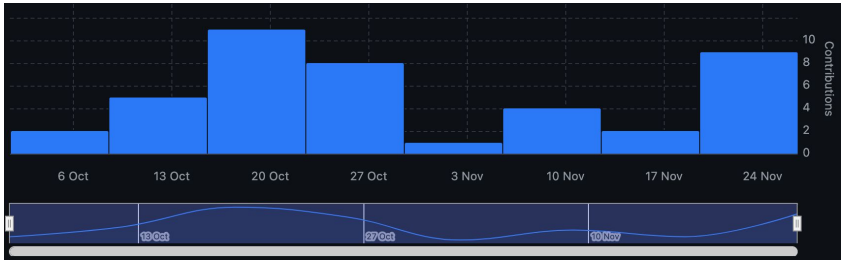
1. Introduction and Basic Functionalities

Inspired by Too Good To Go

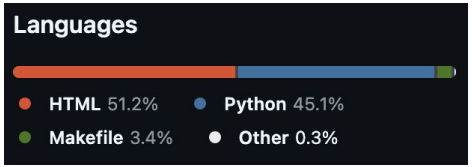
Goal: Reduce food waste by connecting sellers with surplus food to customers



Home page.



Commits over time: weekly from 6 Oct to 24 Nov.



Programming languages used.

1. Introduction and Basic Functionalities

User Registration and Login

- Custom forms
- Password complexity and email validation
- Auto-login after signup
- New users start as Customer, request to be a Seller via email

Product Management

- Sellers: Create / Edit / Delete own products (name, description, price, stock, image)
- Everyone: View all products on products page
- Search bar (by name, description, seller name)

Profile Management

- Edit personal info, phone, birthday (min age 16)
- Upload / change profile picture
- Delete own account

Purchase Flow

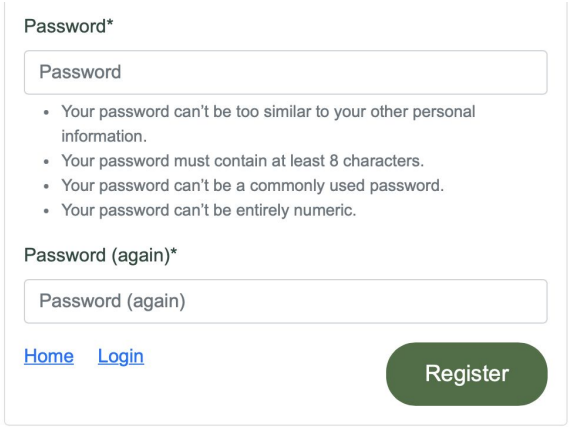
- Customers can “buy” products
- Stock automatically updated

Admin Panel

- Change user roles and delete accounts
- Statistics dashboard (revenue, top sellers, low stock alerts…)

Policies (Password, username, email, birthdate, phone)

- Password: Size, difficulty
- Username, first name, last name: Size
- Email: Valid email address
- Birthdate: At least 16 years old
- Phone: Valid phone number



Register page. Password.

Password protection in the database

- PBKDF2 with SHA256 hash
- By Default (Recommended by NIST)
- Iterations: Slow down attackers
- Salt: Protect against rainbow attacks
- Other options:
  - Pbkdf2_sha1
  - Argon2
  - Scrypt

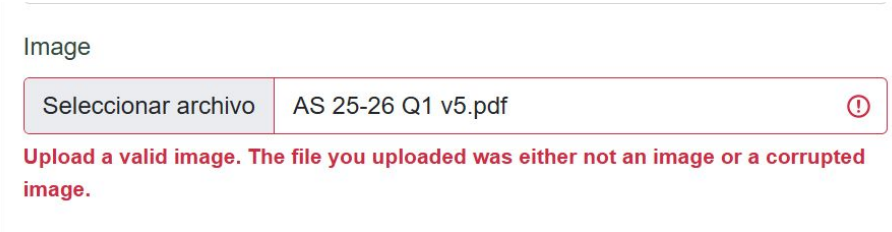
Admin page. Example of the password of a user.

Secure file uploads (only JPG and PNG files and a size not bigger than 5Mb)

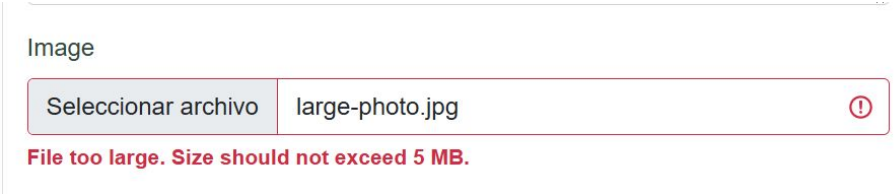
Mitigated risks

- File upload attacks
- MIME type spoofing
- Oversized files causing DoS
- Improved UX with clear messages

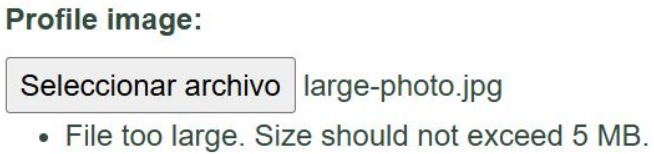
The validations are implemented also in the profile where the user can have a profile photo.



Products page. Invalid file type example.



Products page. File exceeding size example.



Profile page. File exceeding size example.

Role-Based Access Control

3 Available Roles
Admin: Full access, can assign/change roles for any user
Customer: End customer (view products, place orders, manage own account)
Seller: Vendor (manage products, manage own account)

Role Flow

- Every new user starts automatically as **Customer**
- Only **Admin** can promote or demote users to **Seller** or **Customer**

Technical Implementation

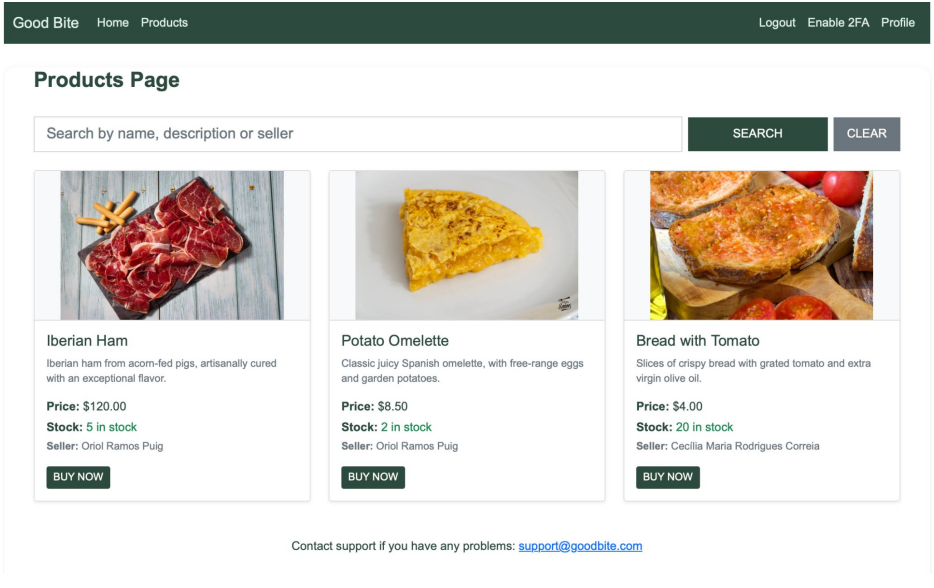
- Roles = Django Groups (Customer, Seller)
- Specific permissions defined per app/model
- Access control using *@permission_required*, *user.has_perm()*
- Role assignment done via Django Admin

Contact us if you want to be a seller: seller@goodbite.com

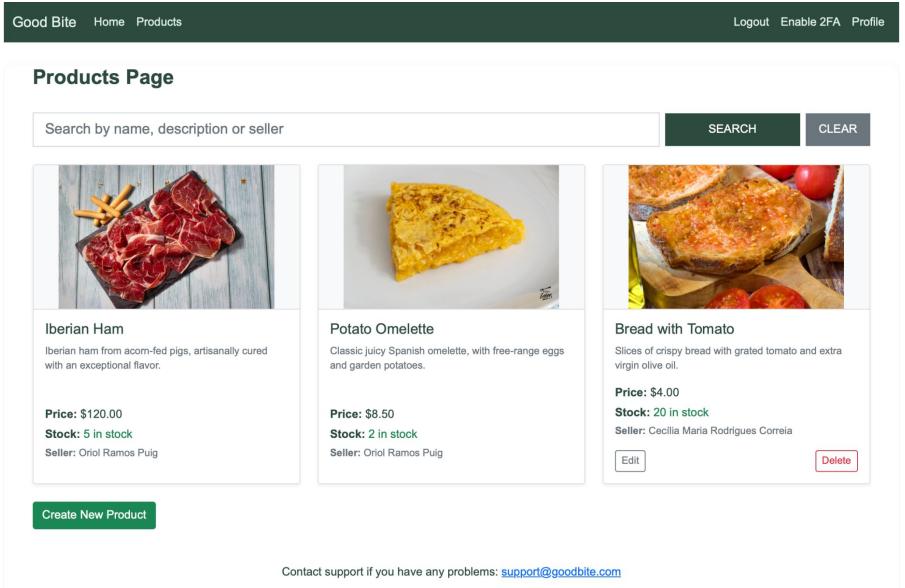
Profile page of a Customer.

Source
Fahim Ahmed. (2024, Sep 25). Role-Based Access Control (RBAC) in Django.

Role-Based Access Control

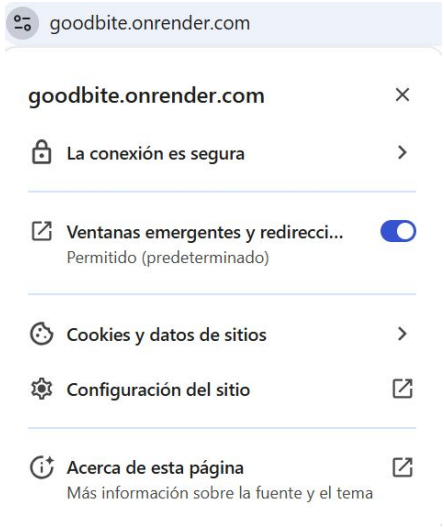


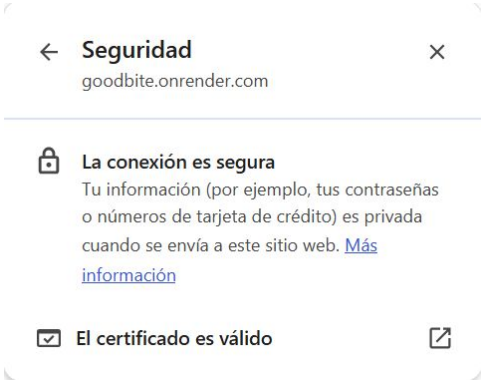
Products page of a Customer.



Products page of a Seller.

Creating and installing an HTTPS server certificate





Render uses **Let's Encrypt** and **Google Trust Services** to issue certificates for your custom domain and automatically renews them before their expiration date.

Source

Render. (n.d.). Fully Managed TLS Certificates – Render Docs. Retrieved from https://render.com/docs/tls
Render. (n.d.). Cloud Application Platform | Render. Retrieved from https://render.com/

MFA

- As an additional layer of security to protect the login process.
- 2FA: user / password + code
- Time based One Time Password: TOTP (RFC6238)
  - Generates a code based on HMAC of a secret and the current time.
  - Temporal code generated by an auth app
- Django-AllAuth.MFA module

Source
https://docs.allauth.org/en/latest/mfa/index.html

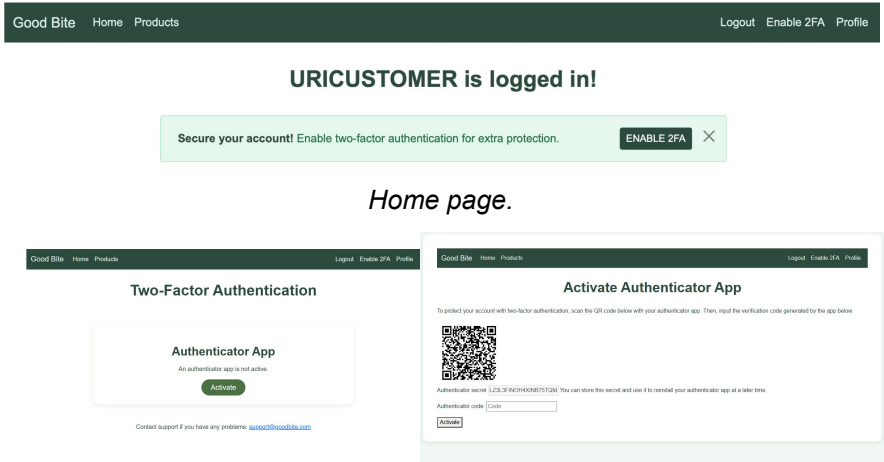
MFA - Flow

New users

- User registration.
- Activate option.
- Select Auth App and scan QR.
- Auth App is enabled to perform 2FA.
- Next login, user is able to login with 2FA.

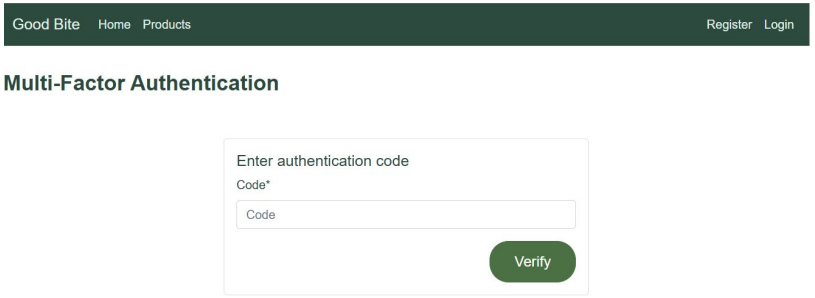
Users

- User log-in
- 2FA page.
- Insert Time based One Time Password.
- User enter to the web page.



Home page.

Enable 2FA page.



Login page.

1st Vulnerabilities Check

Tools: pip-audit, Bandit

Findings:

- Hardcoded SECRET_KEY (low)
- App exposed via 0.0.0.0 (Medium)

Outcome: Minimal code review

2nd Vulnerabilities Check ASVS Review

Focus on OWASP ASVS 4.0.3 L1

Findings:

- Invalid data accepted (email,phone,birthdate)
- Long input broke UI (>500 chars)
- Weak MFA (no reauth)
- DEBUG=True & unsafe ALLOWED_HOSTS

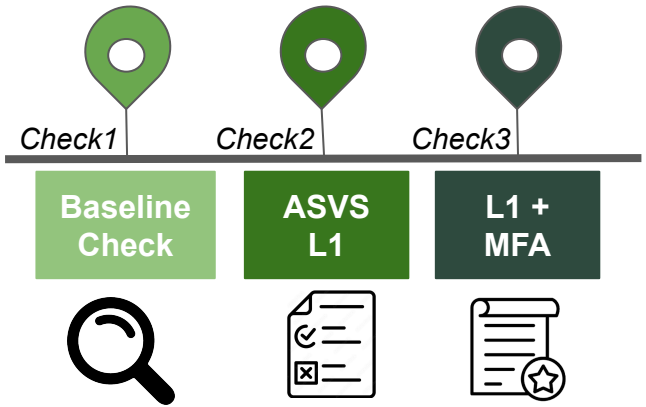
Outcome: Several issues NOT OK.

3rd Vulnerabilities Check - Fix Validation + MFA

All previous issues fixed

- Robust input validation
- File upload protections working
- MFA functioning correctly
- RBAC & IDOR confirmed secure
- ALLOWED_HOSTS restricted
- DEBUG controlled via env variables

Outcome: Stable L1 compliance with targeted L2 improvements.



Sources
Security linter for Python. https://bandit.readthedocs.io
Dependency vulnerability auditor. https://pypi.org/project/pip-audit
Application Security Verification Standard 4.0.3-es.pdf

4. Problems Found

MFA

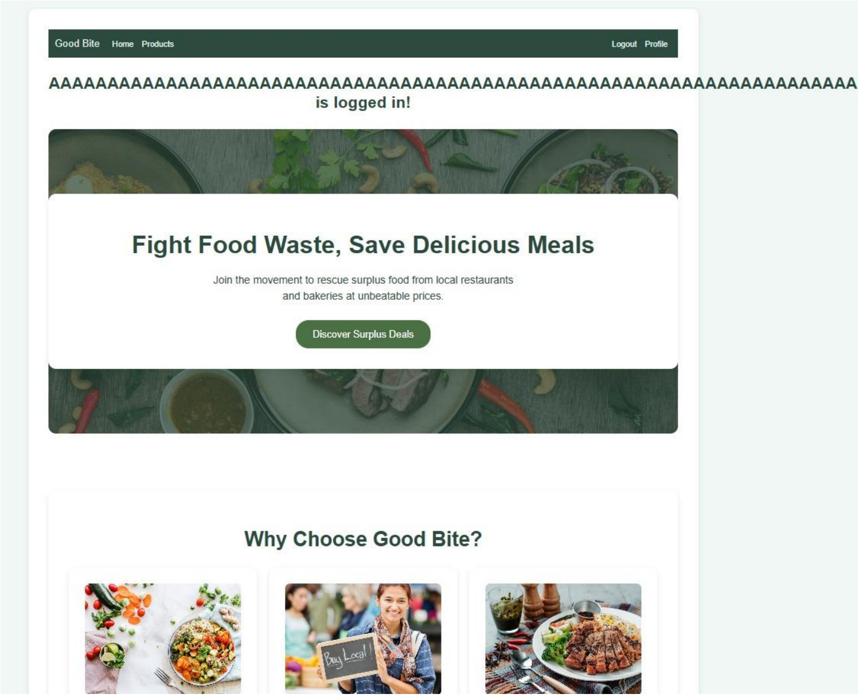
- The MFA library (allauth) removed the original registration form from our previous implementation.
- Challenging to maintain our desired user flow because of library constraints.
- Difficulties encountered when applying custom template styling to default MFA pages.

Certificate

- Initial attempt with a simple self-signed certificate resulted in a browser warning page.
- Created a Certificate Authority and signed the certificate, but the warnings persisted.
- Successfully resolved by deploying to production environment via Render.com.

Layout

- Lack of character limits on username, first name, and last name fields caused layout inconsistencies.



Home page with exceeded layout.

5. Possible Improvements and Conclusion

Possible improvements

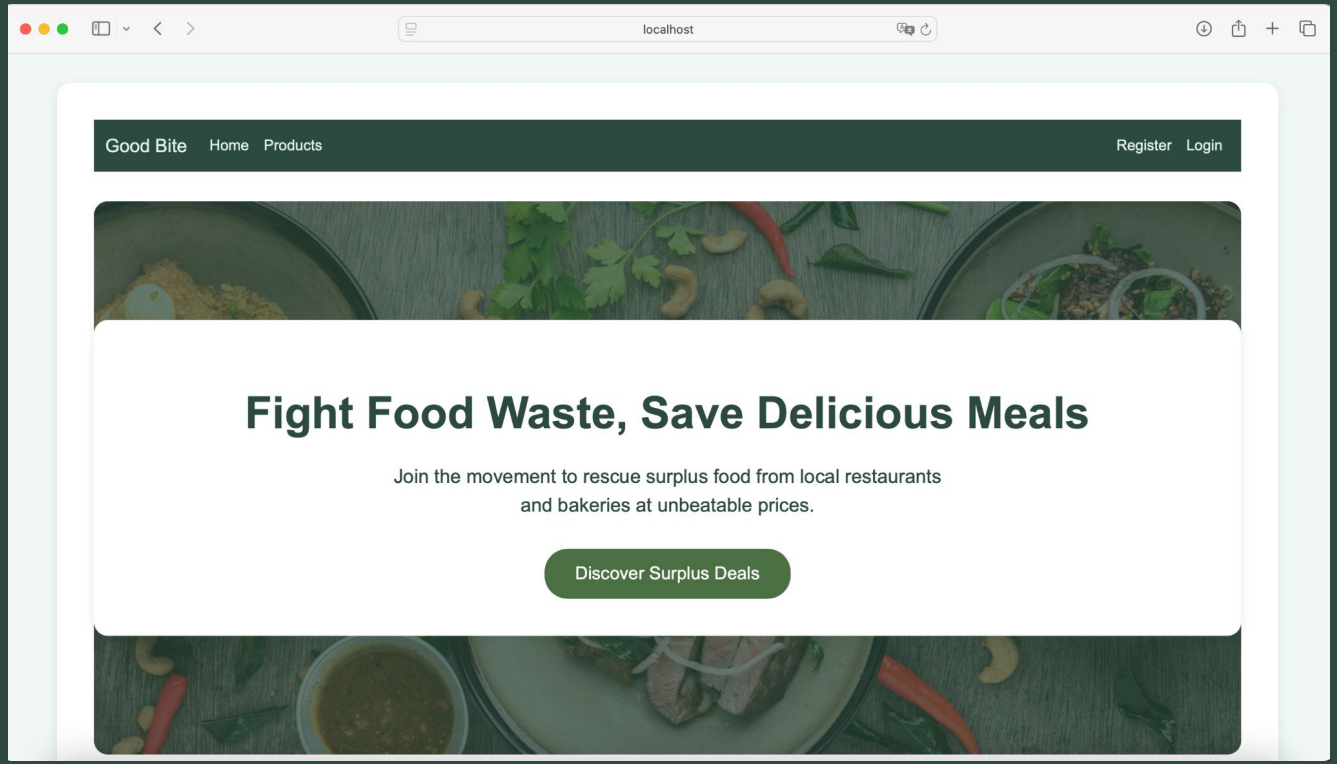
- Implement content encryption, beginning with user profile photos.
- Expand payment options beyond the current "cash only" model to include card payments.
- Enhance product details with additional features such as allergen tags.
- Introduce filtering capabilities to complement the existing search functionality
- Add MFA requirements for high-risk actions.

Conclusions

- Using Django combined with Docker for environment consistency enabled more agile development.
- Integrating OWASP ASVS 4.0.3 throughout the development process allowed early vulnerability identification and ensured security was embedded from the ground up.
- Regular audit cycles facilitated the detection and resolution of coding and configuration errors, including validation gaps and insecure default settings.
- Resulting in a secure web application designed to combat food waste through controlled food sharing and comprehensive traceability.

UNIVERSITAT POLITÈCNICA DE CATALUNYA BARCELONATECH

Thank you!



Anna Melkumyan Canosa · Cecília Maria Rodrigues Correia · Irene Cerván Barriga · Johanna Nuñez Diaz · Oriol Ramos Puig