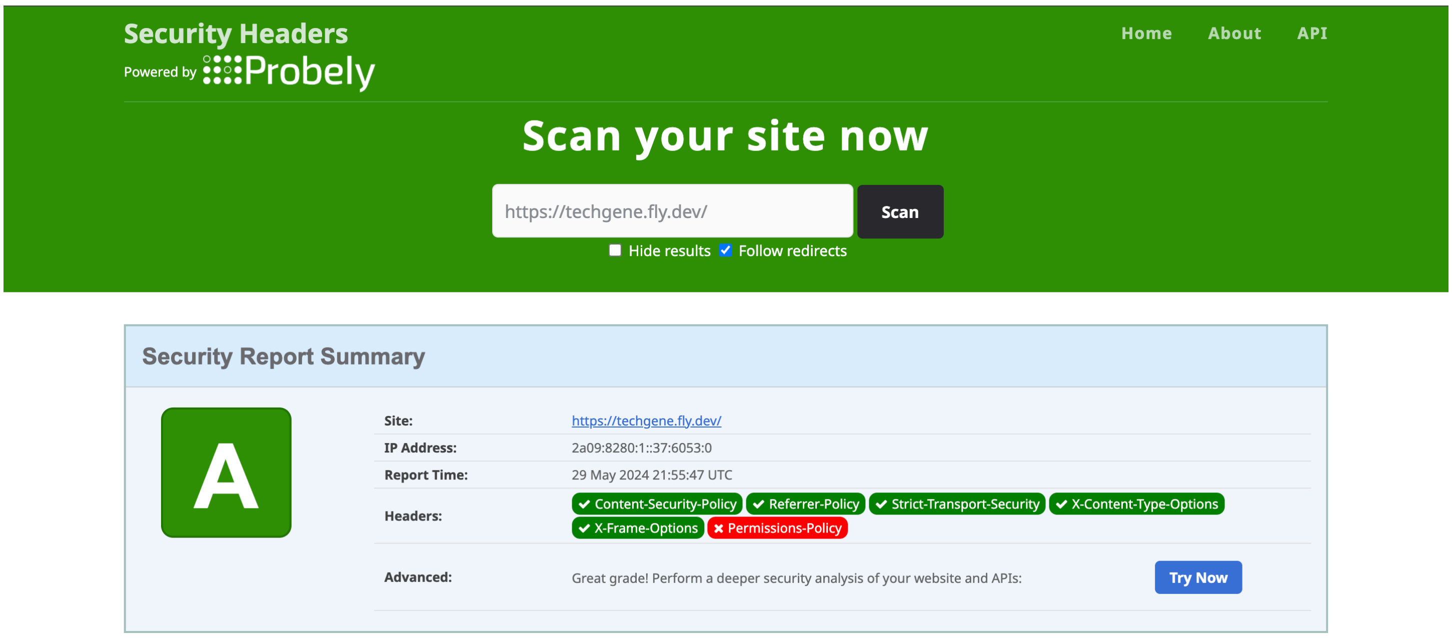
Planning Document

Live Demo

You can test the live API deployed on Fly.io at:

https://techgene.fly.dev



Security Report Summary

A

Site:	https://techgene.fly.dev/
IP Address:	2a09:8280:1::37:6053:0
Report Time:	29 May 2024 21:55:47 UTC
Headers:	✓ Content-Security-Policy ✓ Referrer-Policy ✓ Strict-Transport-Security ✓ X-Content-Type-Options ✓ X-Frame-Options ✗ Permissions-Policy
Advanced:	Great grade! Perform a deeper security analysis of your website and APIs: Try Now

Task

Develop a backend service for a wishlist feature. Users should be able to:

- add products to their wishlist
- remove products
- view their wishlist, and
- ideally share it with others

> *without having to sign up on the platform.*

Tech Stack

- Nx
- TypeScript: Instead of plain JavaScript for enhanced code maintainability and to catch errors early on.
- Express
- Mongoose: For streamlined MongoDB interactions and schema validation..
- Helmet, cors, CSRF: To enhance security.
- WInston: Logging.
- JWT: Authentication
- Redis: For fast caching of wishlist data, especially for high traffic

Data Model

- [x] Set up MongoDB data model using Mongoose as follows

TypeScript

```typescript
const Wishlists = new Schema(
  {
    sessionId: {
      type: String,
      required: true,
    },
    items: [
      {
        product: { type: Schema.Types.ObjectId, required: true },
        addedAt: { type: Date, default: Date.now },
      },
    ],
    isPrivate: {
      type: Boolean,
      default: false,
    },
  },
  {
    timestamps: true,
  }
);

const Products = new Schema(
  {
    name: {
      type: String,
      required: true,
    },
    image: {
      type: String,
      default:
        'https://images.pexels.com/photos/6208084/pexels-photo-6208084.jp
    },
    price: {
      type: Number,
      required: true,
    },
    isDeleted: {
      type: Boolean,
      default: false,
    },
    inStock: {
      type: Boolean,
      default: true,
    },
  },
  {
    timestamps: true,
  }
);
```

API Endpoints

The frontend can use `axios`, `fetch` or **Postman** to interact with these API endpoints. Postman is used here for testing.

Public Routes

- [x] `POST /wishlists`: Creates a new wishlist and returns a `wishlistId` and a JWT `token` (from the auto-generated sessionId).

  > *The JWT authorization is important to prevent other guest users from making changes to this particular guest user's wishlist. The JWT token is to be stored in the user's localStorage and returned in subsequent requests to protected routes to verify this particular guest user.*

- [x] `GET /wishlists/:id`: Retrieves a wishlist, with proper labelling for `isDeleted` and `inStock` products for proper labelling on the FE

Protected Routes (requires a valid Bearer token)

- [x] `DELETE /wishlists/:id`: Deletes a wishlist.
- [x] `PATCH /wishlists/:id/privacy`: Toggles the privacy status of a wishlist.

  > *If a wishlist is private, it cannot be viewed by other guest users.*

- [x] `POST /wishlists/:id/add-item`: Adds a product to a wishlist
- [x] `DELETE /wishlists/:id/remove-item/:productId`: Remove a product from a wishlist.

Admin Routes (publicly available for test purposes)

- [x] `POST /products`: Adds a product to the DB
- [x] `DELETE /products/:productId`: Sets the `isDeleted` status of the product to true

  > *Use a cron/batch job to delete products with `isDeleted` status from the DB and from wishlists at a later time for faster front-end response*

- [x] `PATCH /products/:productId/availability`: Toggles the `inStock` status of a product

Key Features

Session Management

Backend

- [x] Generate unique `sessionId` and `wishlistId`.
- [x] Store wishlist data with `sessionId` and `wishlistId`.
- [x] Create a JWT token from the `sessionId`.
- [x] Return `wishlistId` and JWT `token` to frontend.
- [x] Retrieve and validate the JWT token for subsequent protected API requests
- [x] If they match, allow the modification. If not, return an error: `403 Forbidden` or `401 Unauthorized`

Frontend (Postman for testing purposes)

- [x] Securely store JWT token in Postman environment variable.
- [x] Include JWT token in the Authorization header(`Bearer <token>`) for protected API requests.
- Handle authentication errors gracefully.

  > *Handled by default by Postman*

Sharing

- [x] Allow users to share their wishlist
- [x] Adding a mechanism to revoke sharing
- [x] Include relevant product details (e.g., name, image, price) fetched from the product database, to provide a more complete user experience.

Security

- [x] Implement CSRF protection, and other security measures using Helmet, and cors.
- [x] Validate user input to prevent injection attacks.

Error Handling

- [x] Provide meaningful error messages and HTTP status codes for client-side handling.

Documentation

- [x] Create clear and comprehensive API documentation using Postman.

Deployment

- [x] Configure `MONGODB_URI` to MongoDB Atlas value
- [x] Deploy to fly.io: https://techgene.fly.dev/

Testing

Done with Postman

Room for Improvement

Sorting and Filtering

- Add endpoints for sorting wishlist items (e.g., by date added, product name) and filtering them (e.g., by product category, price range).

```yaml
GET /wishlist/:wishlistId?sort=dateAdded&filter=category:electronics
```

Scalability

- Consider Redis for caching and explore MongoDB sharding for potential future scaling

Security Improvement

- [x] Use HTTPS to secure data transmission.

> *Test API security [here]()*

Notifications

- Notifications when a wishlisted item goes on sale or is back in stock.
- Notifications for guest user to sign up so their wishlist doesn't get lost.

Model

- [x] Make wishlists not return `sessionId` at the Model level instead of using `select('-sessionId')`
- [x] Add indexes to the `productId` fields in both Products and Wishlists models to improve query performance.
- Cron/batch job to delete products with `isDeleted` status from the DB and from wishlists at a later time for faster front-end response
- Cron/batch jobs to delete guest wishlists that have not been accessed after a while

Testing Improvements

- Include unit and e2e testing

Error Handling Improvements

- [x] Implement standard error handling middleware to catch errors gracefully and return appropriate responses to the client.

Multiple Wishlists