

# Mehrez Al-Ayari

 Mehrez45 |  Mehrez Al-Ayari |  mehrezayari706@outlook.com

## SUMMARY

---

Computer Science student at the **University of Surrey** specialising in backend development and applied machine learning. Experienced in designing scalable REST APIs and database-driven systems using **Java** and **Python**. Strong understanding of core algorithms, data structures, and mathematical modelling through hands-on development and research projects.

## EDUCATION

---

2024 - present	<b>BSc Computer Science at University of Surrey</b> Web & Database Systems (80), Operating Systems (75), Software Engineering (72) <b>Overall</b> first year grade: 68.75%
2021 - 2023	<b>John Madejski Academy</b> <b>A-levels:</b> Mathematics (A), Sociology (A), Physical Education (B)
2016 - 2021	<b>Lampton Academy</b> <b>GCSEs:</b> Combined Science (9,8), Business (9), Physical Education (9), Mathematics (8), English Language (8), English Literature (7), Religious Studies (7), History (7), Film Studies (7)

## PROJECTS

---

### Dottify — Django (Python, Django REST Framework, SQLite)

Developed a Spotify-style music web application with a Django backend, featuring structured domain models, RESTful APIs, authentication, and comprehensive automated testing.

- Designed and implemented relational **models** for users, artists, albums, songs, and playlists using the Django ORM.
- Built RESTful endpoints using **APIView** and **class-based views** to support playlist creation, song management, and playback-related operations.
- Integrated secure user **authentication and authorisation** using Django's built-in auth system and token-based API access.
- Connected frontend templates to backend views, enabling dynamic HTML rendering and consistent database-driven UI behaviour.
- Implemented **comprehensive automated testing** using Django's **TestCase** and **APIClient**, covering models, API endpoints, HTML views, authentication flows, and edge cases to ensure correctness and prevent regressions.
- Configured **SQLite** for development, with schema design compatible with migration to PostgreSQL.

### Java Movie Database Backend — Java (Javalin, SQLite, JUnit, Mockito)

Designed and implemented a layered REST backend serving IMDb-style movie data with a clean separation between routing, service, and persistence layers.

- Built REST endpoints using **Javalin** (`/movies`, `/movies/{id}`, `/search`) returning JSON responses with proper HTTP status handling.
- Modelled a relational schema in **SQLite** and implemented efficient read-only queries using indexed fields for fast lookup.
- Wrote extensive **unit and service-level tests** with **JUnit** and **Mockito**, mocking database access to isolate business logic and ensure deterministic API behaviour.

## CNN Classification on Polygon Images — Python (PyTorch, NumPy, Matplotlib)

Developed a **compact Convolutional Neural Network (CNN)** to classify synthetic polygon images under strict data and training constraints.

- Limited the model to **150k parameters** due to small training/validation sets and short epoch budgets.
- Used small convolutional stacks, batch normalisation, dropout, + **GAP** to mitigate overfitting.
- Trained via a custom **training loop** using Adam/SGD optimisers with cross-entropy loss and epoch-wise loss and accuracy tracking.
- Balanced model capacity and regularisation to improve generalisation under limited data.
- Implemented end-to-end CPU-based training and evaluation without reliance on high-level training abstractions.

## K-means Clustering & Multiclass Logistic Regression — Python (NumPy, Pandas ...)

- Designed and implemented a fully custom **K-means clustering pipeline** for polygon images, supported by hand-crafted geometric feature extraction.
- Engineered invariant geometric features (centroid normalisation, radial distance distributions, edge-length and angle statistics) to enable meaningful clustering under rigid transformations and noise.
- Implemented **multiclass Logistic Regression** from scratch using gradient-based optimisation and cross-entropy loss as a supervised baseline for comparison.
- Emphasised first-principles understanding by implementing all algorithms directly with NumPy, without relying on pre-built machine learning libraries.

## Linear Regression Implementation from Scratch — Python (NumPy)

[Link to Demo](#)

Developed a Linear Regression model from scratch, implementing key components **including:**

- **Optimisation:** Gradient Descent for iterative weight and bias updates
- **Analytical Solution:** Normal Equation for direct computation of optimal weights
- **Loss Functions:** Mean Squared Error (MSE) and Mean Absolute Error (MAE)

This reinforced in-depth understanding of machine learning fundamentals by building all components from the ground up. The model was tested on synthetic and real datasets, achieving accurate predictions without using pre-built machine learning libraries.

## SKILLS

---

**Programming Languages**    Python, Java, C++, JavaScript, SQL

**Frameworks & Tools**    Django, Django REST Framework, Javalin, NumPy, Pandas, Scikit-learn, Git, JUnit, Mockito

**Databases**    SQLite, MySQL

**Concepts**    Object-Oriented Programming, Data Structures & Algorithms, RESTful API Design, Machine Learning Fundamentals, Software Testing, Version Control

**Operating Systems**    macOS, Linux, Windows