Prisoner Data Analysis and Visualisation

JACK GILMORE

The brief

"You are required to develop a Python application that processes prisoner data, provides a REST API for accessing the data, and displays the data as interactive charts in a web browser."

Milestones

Data Processing and Analysis

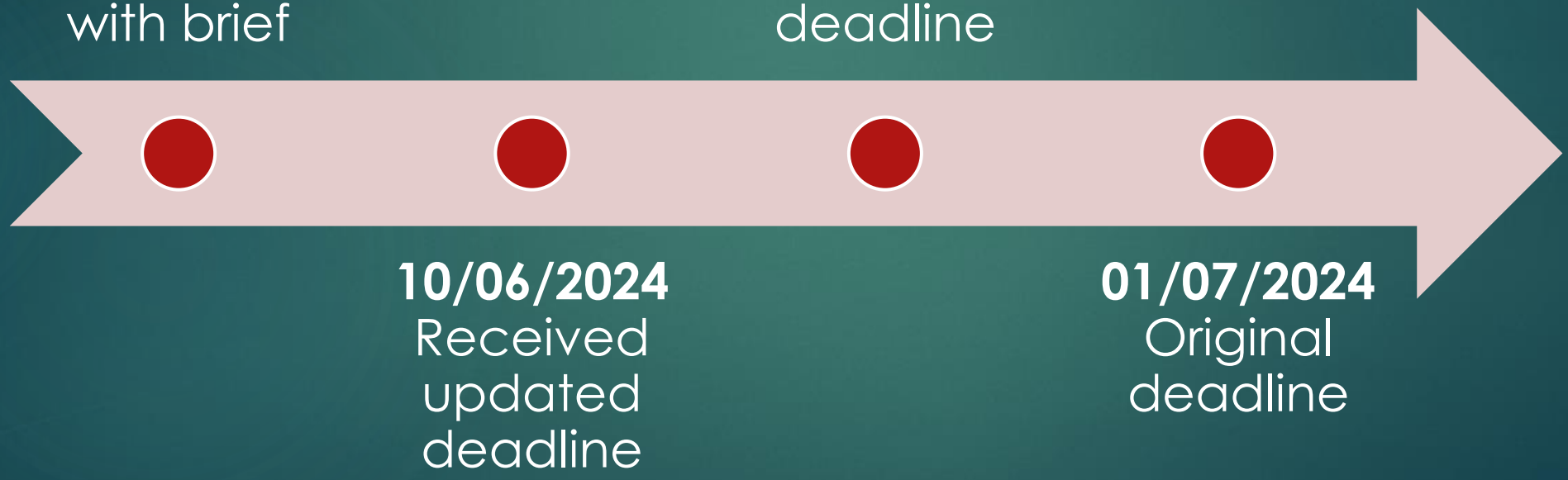
API Development

Front-End Development

Security and Best Practices

Deployment (stretch goal)

Timeline


07/06/2024
Initial invite with brief
10/06/2024
Received updated deadline
19/06/2024
New deadline
01/07/2024
Original deadline

Research

- Scottish Prison Population Statistics 2022-23
  - https://www.gov.scot/publications/scottish-prison-population-statistics-2022-23/
- Scottish Prisons Interactive Analysis Tool Shiny App
  - https://scotland.shinyapps.io/sg-prison-population-statistics/
- Scottish prison population statistics technical manual
  - https://www.gov.scot/publications/scottish-prison-population-statistics/
- Scottish Government Design System
  - https://designsystem.gov.scot/

Strategy

- Task management
- Minimum Viable Product
- Clear acceptance criteria
- Architectural decisions
- In a real-life scenario, constant feedback is beneficial

Architectural decision review

- Option 1: MySQL
  - Pros:
    - Mature and widely used, especially in web applications.
    - Good support for transactions and complex queries.
    - Scalability for large datasets and high traffic.
  - Cons:
    - Requires setup and maintenance of a separate infrastructure.
    - May be overkill for smaller applications?
- Option 2: SQLite ✅
  - Pros:
    - Serverless and self-contained; no separate server setup needed.
    - Simple to deploy and manage; ideal for small to medium applications.
    - Good performance for read-heavy operations.
  - Cons:
    - Limited concurrency compared to client-server databases like MySQL.
    - Less suitable for write-heavy applications or large datasets.
    - Lack of advanced features like stored procedures and user management.

Execution - database

prisoner_id	name	age	gender	crime	sentence_years	prison
integer	string	integer	character	string	integer	string
1	John Doe	34	M	Murder	25	Barnard Castle
2	Jane Smith	29	F	Theft	5	Edinburgh
3	Bob Johnson	42	M	Fraud	10	Glasgow

Execution - database

Execution

Load data and analyse

- load_data.py (PyMuPDF)
- analysis.py (Pandas)
- database.py (SQLAlchemy)

API development

- database.py (SQLAlchemy)
- main.py (Fast API)
- Swagger

Front-End development

- index.html
- chart.js
- custom-charts.js

Security considerations

- Storage of data at rest
- Database authentication
- API authentication
- Input validation and sanitisation
- Access control – code and app
- Auditing

Lessons learned

- FastAPI – routing ordering matters with static files
  - Implement stronger test suite to identify regressions
- SQLite – Foreign keys not enforced by default
  - Trust ORMs, but don't rely on defaults and consult documentation

What would I do if I had more time?

- JS and CSS package management via NPM
- Bundle and minify assets
- Extend tests
  - Database integration tests
  - Model function tests
  - System tests in the web browser
- More validation and checks on data load

What would I in a real production scenario?

- Centralised or more sophisticated API authentication
- Database infrastructure
- Dev / Test / UAT / Staging environments
- Statistical anonymisation
- Monitoring, alerting and analytics
- Infrastructure as code – aids with disaster recovery

Questions?