Full-Stack Code Assignment – F1 World Champions

1. Goal

Build and implement a production-ready, end-to-end web/mobile application that:

- Shows Formula 1 World Champions from 2005 to the present year in an attractive Single Page Application (SPA) or Mobile Application.
- Persists race-winner data in a database and exposes it through your own backend API.
- Ships with a fully automated CI/CD pipeline.
- Runs backend and DB locally with a single docker compose up command.

The exercise lets you demonstrate full-stack skills: API & data modelling, secure coding, automated testing, DevOps, and clean, reusable front-end code.

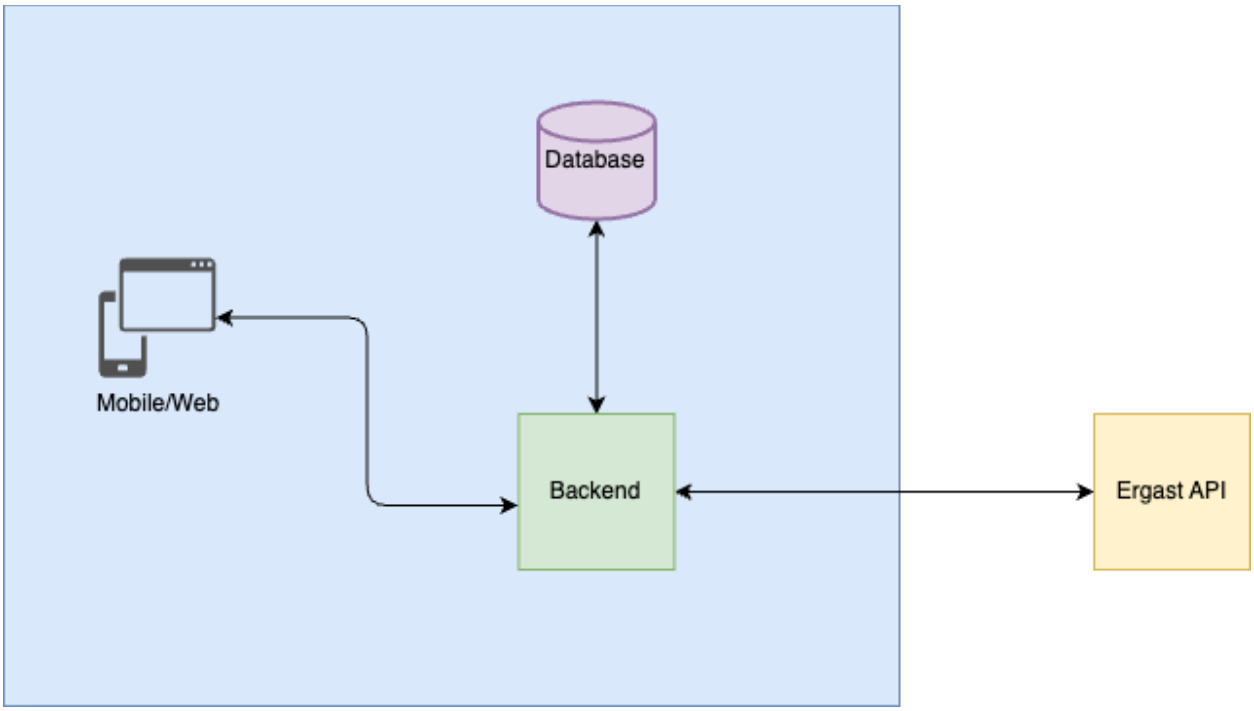
2. Functional Requirements

1. **Season list** - Display each season's World Champion (2005 to present).
2. **Race winners** - Clicking a season reveals all grand-prix winners for that year.
3. **Highlight champions** - In the race list, visually highlight the rows where the winner is also that season's champion.
4. **Persist winners** - On first request, fetch data from the public Ergast Developer API (https://api.jolpi.ca/ergast/), then store or update it in your own database. Subsequent requests must be served from your backend.
5. **Graceful errors & loading states** - Handle slow networks, API errors, and empty states.

3. Non-Functional Requirements

Layer	Mandatory	Nice to have
Frontend	SPA with React, Vue, Angular or Svelte. 100 % typed (TypeScript). State handled with a modern approach. **OR** Native Mobile with Swift/Kotlin. Tests (unit + component).	SSR/SSG (Next.js/Nuxt), Lighthouse score ≥ 90.
Backend	Language & framework of your choice (Node/Express, Spring Boot, etc.). REST **or** GraphQL endpoint(s). Proper layering (domain / service / controller). API contract documented (OpenAPI). Unit tests.	Caching layer (e.g. Redis). Async job to refresh seasons weekly after every race.
Database	Relational (PostgreSQL, MySQL) or NoSQL (Mongo). Add sensible indexes & constraints.	Seed script; containerised admin tool (pgAdmin, Mongo Express).
CI/CD	GitHub Actions, GitLab CI, Butbucket Pipelines, TravicCI, CircleCI or any free tier CI service. Stages: *install → lint → test → build*. Reject on test failure.	Automatic deploy to free tier platform (Render, Railway, Fly.io, or your own). Security: Dependency scan via **CodeQL/Snyk/Trivy**. Reject on scan failure. Docker image pushed to public registry.
Containerisation	Multi-stage Dockerfiles. Single docker-compose.yml starting **backend**, **database**. Healthchecks & environment variables.	Makefile targeting common tasks.

4. Suggested Architecture



- The backend is the only component talking to Ergast API. It normalises responses and saves winners.
- Frontend consumes your API only – *no direct calls* to Ergast.
- Optional: add a caching layer between backend and DB for faster cold starts.

5. Deliverables

1. **Git repository** (GitHub/GitLab/BitBucket) containing:
    a. frontend/ – SPA/Mobile source code.
    b. backend/ – API service.
    c. infrastructure/ – Dockerfiles, docker-compose.yml.
    d. .github/workflows/ *or* .gitlab-ci.yml – CI/CD pipeline.
2. **README.md** explaining:
    a. High-level architecture & trade-offs.
    b. How to run locally (docker compose up), run tests, and trigger pipeline.
    c. API contract / schema.
3. **Automated tests** (≥ 70 % coverage for critical logic).

4. Diagrams or screenshots where useful.

6. Evaluation Criteria

1. **Code Quality & Architecture** - readability, modularity, separation of concerns.
2. **Testing** - coverage depth and strategic choice of tests.
3. **Security & Reliability** - secret management, input validation, dependency hygiene.
4. **CI/CD Maturity** - pipeline completeness, speed, and gating effectiveness.
5. **User Experience** - responsive, accessible UI, clean design.
6. **Documentation** - clarity of README and inline comments.

7. Submission

- Share the repository **link** and optionally a published URL.
- Ensure reviewers can run: git clone ... && cd f1-app && docker compose up (to start the BE and DB and optionally the WebFE)

8. FAQ

What technologies can I use?

Any technologies you are comfortable with. Recommendation is React/Angular for web frontend, NodeJS/Spring Boot for backend and Flutter/ReactNative/Swift/Kotlin for mobile.

What is the time frame for completion?

You will have until **end of day on June 2, 2025**, to complete the assignment.

What if I am working on a project at the same time?

If you are working on a project and not able to complete it within the given time frame, we will extend it on a case-by-case basis.

How do I log my time if I am working on a project and doing the code assignment?

A project code for training will be provided.

What accounts will I need to complete it?

- Github/Gitlab/Bitbucket
- Docker

Can I still consider a settlement offer after completing the first assessment?

Yes. At any point during the process, you can request a draft settlement offer for your own consideration.

When will we get feedback on the assessment?

By the 15th of June the evaluations will be completed, and we will start providing feedback.

What happens after the assignment is submitted?

After you have submitted your assignment, it will be reviewed and a call with be scheduled with you to discuss the assignment. After that discussion, you will be informed of the outcome.

Is it possible to skip the first evaluation and do training first?

No, even if you feel you need additional training before the first round, it is recommended to do the first assessment to allow you to understand what the requirements are.

Can I do web frontend if I am a mobile engineer and vice versa?

Yes, as a Frontend engineer, you are welcome to create a mobile app using flutter or react native, and as a mobile engineer you are welcome to create a web app.

What happens if I cannot complete the assignment in time?

You will be graded on the parts that have been completed.

Can I use AI Tools to support the development of the assignment?

Yes.

9. Additional Notes

- Good to remember: the new role will be a full stack role, meaning that you could be asked to join any full stack project in the future. Please keep this in mind when deciding what technologies to use.
- You can use your own personal git repository or raise a request with Ask Genie for a Hexaware gitlab repository.
- If you have additional time, please feel free to add extra features such as API versioning. Anything that would not bloat the codebase too much.
- Feel free to reach out to me at any time for additional information or clarification.
- Remember: we have a whole engineering department with extensive knowledge, please feel free to reach out to the BE/DevOps team if you would like help on a certain topic.
- We have reviewed the assignment in quite some detail internally, however if you spot an error or a blocker of some kind, please reach out immediately so that we can fix it.