Attackme – Technical Documentation

Participants:

- Ray Dotrop
- Kerian Douidi
- Raphael Dott

Initial Role Assignment:

- Temporary Project Manager: Ray Dotrop
- Backend/API: All members
- Frontend/UI: All members

Team Standards:

- Communication: Private Discord server with task-based channels
- Task Management: Trello board
- Decision Making: Simple majority vote, documented in shared Google Doc

---

🧠 1. Research & Brainstorming

Sources of Inspiration:

- Platforms like Root-Me, TryHackMe, HackTheBox
- Capture The Flag (CTF) competitions
- Common issues faced when learning cybersecurity

Brainstorming Methods:

Mind Mapping Topics:

- Web Security
- Networking
- Cryptography

Key Features Identified:

- Authentication
- Dashboard
- Progress Tracking
- Scoring
- Flag Submission

SCAMPER Method Highlights:

- Substitute complex CTFs with simpler, beginner-friendly ones
- Adapt a badge system to track progress
- Eliminate need for VM hosting (initially)

How Might We Questions:

- How might we help beginners progress in cybersecurity?
- How might we make challenges engaging and accessible?

---

2. Idea Evaluation & MVP Selection

Final Idea:
Beginner-friendly cybersecurity challenge platform

Why this choice?

- Balanced feasibility and impact
- Technically possible for a small team
- Scalable to add VMs, Docker, or tournaments later
- A meaningful addition to portfolios

Target Audience:

- Computer science students
- Career switchers entering cybersecurity
- Curious learners without prior experience

Key MVP Features:

- Secure user authentication
- Challenge dashboard
- Flag submission system
- Scoring and leaderboard
- User profile with progress tracking

Out of Scope (for MVP):

- Hosting full virtual machines
- Multiplayer/real-time features
- Advanced DevOps integrations

🎯 3. SMART Objectives

- Develop a basic challenge platform with at least 2 categories by the end of Week 10
- Implement authentication, scoring, and profile tracking within 6 weeks
- Ensure platform usability and collect feedback from 5+ external users before final delivery

---

4. User Stories (Prioritized Using MoSCoW)

Must Have

- As a beginner in cybersecurity, I want to solve categorized challenges so that I can build foundational skills step-by-step.
- As a user, I want to register and log in securely so that I can track my progress and earn points.
- As a user, I want to submit flags for challenges so that I can earn points and mark them as completed.

Should Have

- As a user, I want to see my progress and badges so that I stay motivated.
- As a user, I want to view a leaderboard so that I can compare my score with others.

Could Have

- As a user, I want to receive suggestions for new challenges so that I can keep learning.

Won’t Have (for MVP)

- As a user, I want to participate in multiplayer competitions so that I can challenge others in real time.

---

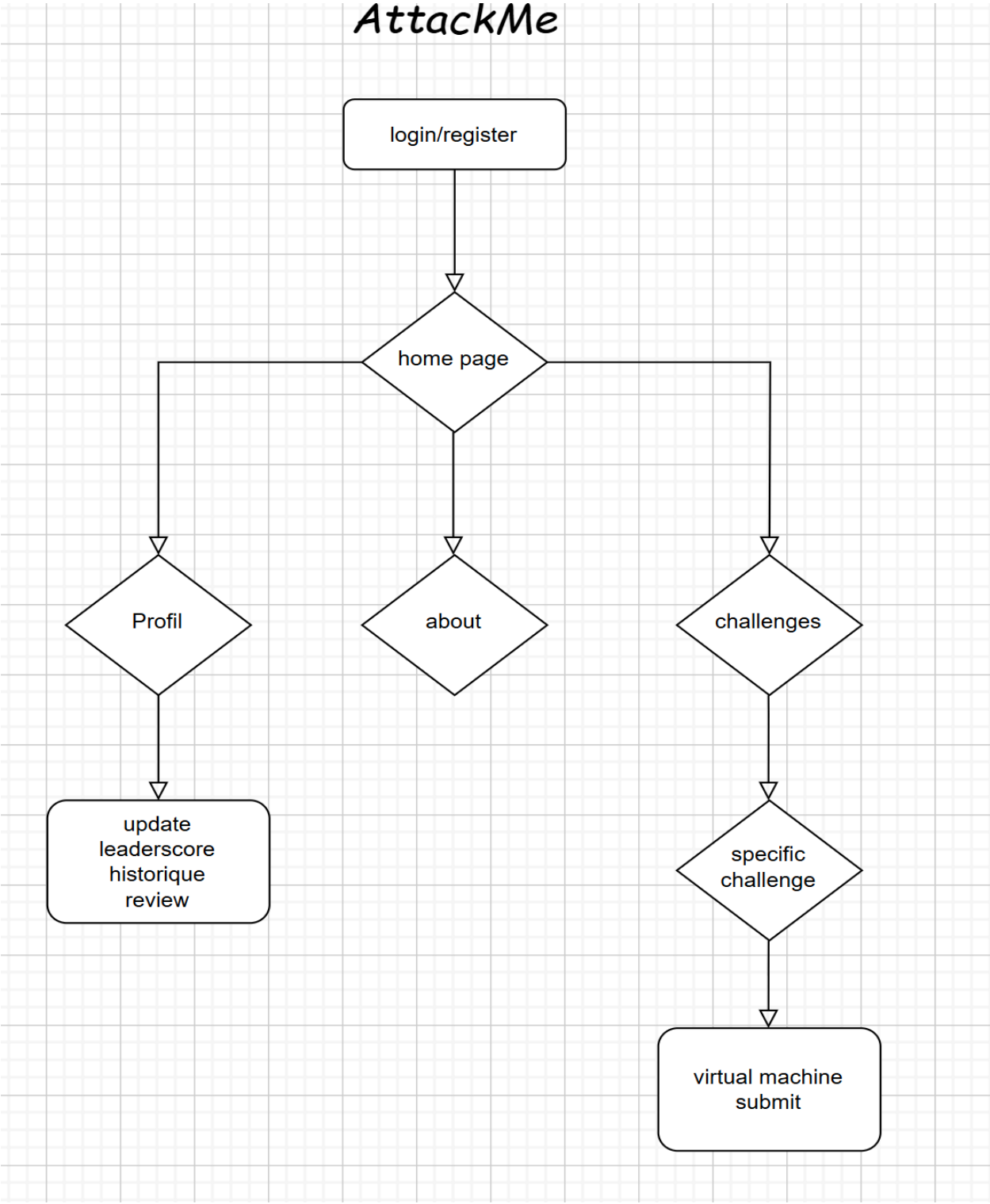
5. Mockups (Optional But Recommended)

Main screens to mock up using Figma, Balsamiq, or Whimsical:

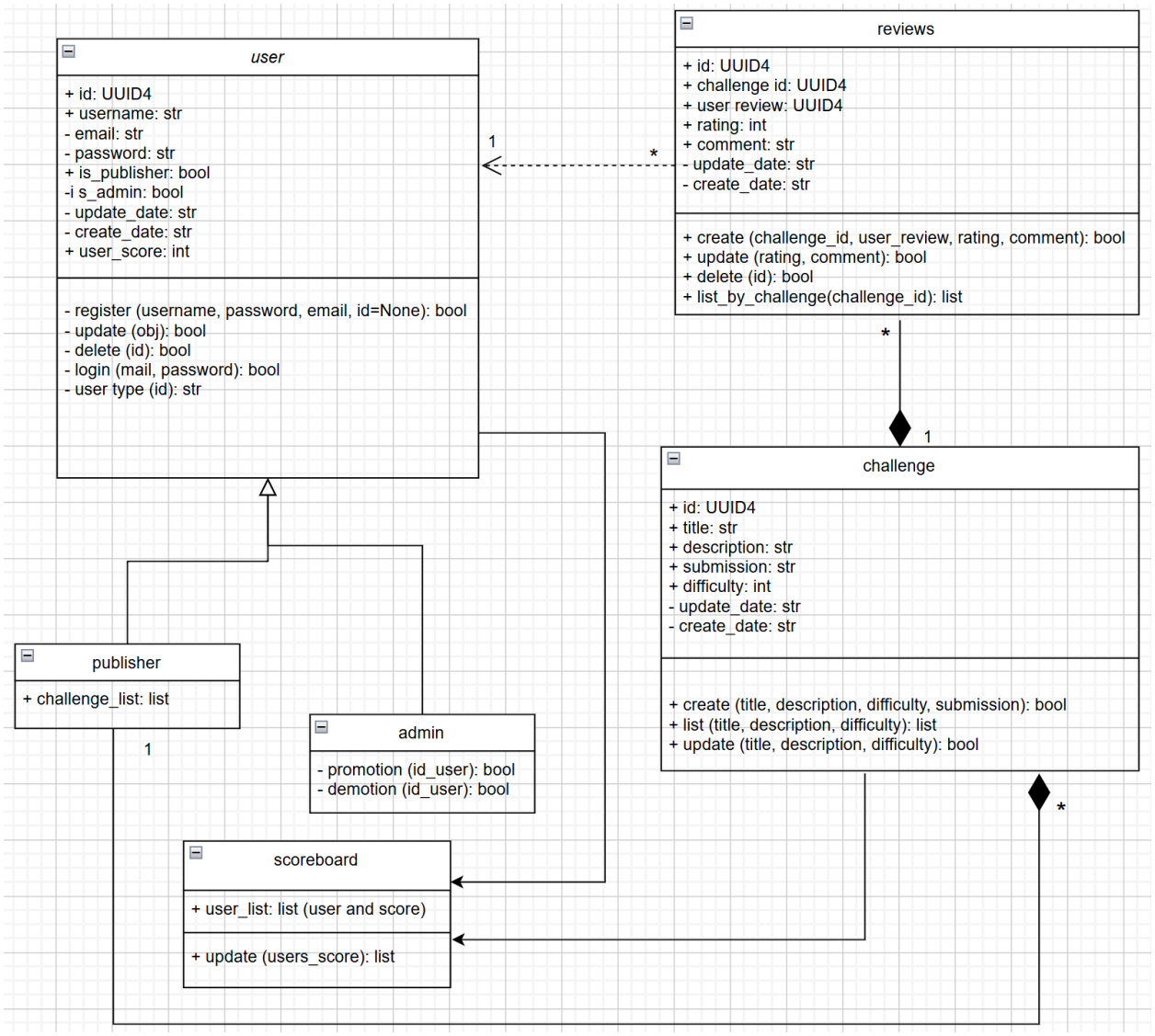
- Login and Register pages
- Dashboard with category filters (Web, Crypto, Network)
- Challenge page with description and flag input
- Profile page showing badges and progress
- Leaderboard page with top scorers

---

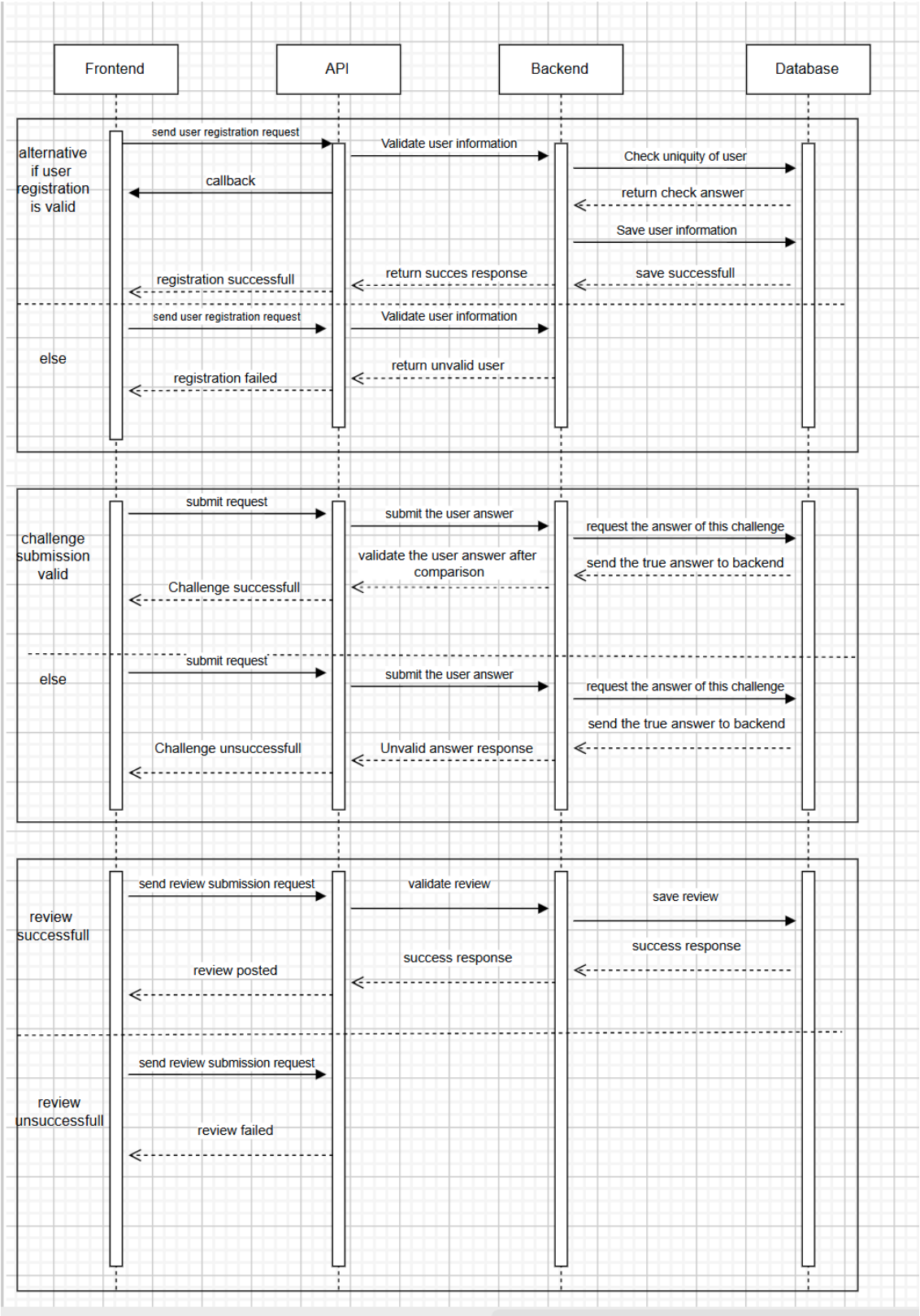
6. System Architecture



7. Classes diagram



8. Séquence diagram



9. API Specifications

External APIs: None planned for MVP

Internal API Endpoints:

POST `/api/register`

- Input: email, password
- Output: success or error

POST `/api/login`

- Input: email, password
- Output: JWT token

GET `/api/challenges`

- Output: List of challenges

GET `/api/challenges/:id`

- Output: Challenge detail

POST `/api/submit`

- Input: challenge ID, flag
- Output: success/failure, score update

GET `/api/profile`

- Output: user stats, badges

GET `/api/leaderboard`

- Output: sorted list of top users

---

10. Source Control and QA Strategy

SCM (Source Control Management):

- GitHub
- Branches: main (production), dev (staging), feature/* for task-specific work
- Code reviews required before merging to main

QA (Quality Assurance):

- Unit testing with Jest
- Integration testing of API endpoints
- Manual testing for user flows (login, submit flag, leaderboard)
