# Sprint 1 Planning Document

1. Sprint Goal  
Set up core backend services and frontend scaffolding to enable client registration, job submission, and developer profile management—interfaces will be stubbed until UI design sprint.

## 2. Sprint Backlog (Total: 18 pts)

| **Epic** | **Story (with Ownership)** | **Acceptance Criteria** | **Subtasks** | **Owner** | **Pts** |
| --- | --- | --- | --- | --- | --- |
| **Account & Onboarding** | Client auth API & routing scaffolding *Owner: You (Lead)* | - **Given** a POST to /api/auth/signup with valid payload  **When** API is called  **Then** user record is created with pendingVerification status and email job is queued in message broker. –  **Given** a GET to /api/auth/verify?token=... **When** token valid  **Then** user status flips to active.  - **Given** a POST to /api/auth/login with valid credentials  **When** API is called  **Then** returns JWT and user metadata (id, name, role). | 1. Define auth service interfaces (signup/login/verify)  2. Implement signup & login controllers  3. Integrate email queue  4. Write verification handler  5. Configure JWT issuance & middleware | Lead | 5 |
| **Profile Management** | Developer profile API & schema *Owner: You (Lead)* | - **Given** a GET to /api/profile/{devId}  **When** profile exists  **Then** returns JSON with skills, experience, availability fields.  - **Given** a PUT to /api/profile/{devId} with valid payload  **When** API is called  **Then** updates DB and returns updated JSON.  - **Given** invalid payload (e.g. availability end before start)  **When** API is called  **Then** returns 400 with validation errors. | 1. Design Profile DB schema/model  2. Build GET/PUT profile endpoints  3. Add validation logic  4. Write unit tests for service layer  5. Add API docs entries | Lead | 5 |
| **Job Submission** | Job request API & message queue *Owner: You (Lead)* | - **Given** a POST to /api/jobs with Title, Description, Skills [], Deadline  **When** API is called  **Then** job records are created with status: pending and event jobCreated published to message broker.  - **Given** GET to /api/jobs?status=pending **When** API is called  **Then** returns the list of pending jobs with all fields.  - **Given** invalid payload  **When** API is called  **Then** returns 422 with field-specific errors. | 1. Define Job schema/model  2. Implements create-job endpoint with validation  3. Publish jobCreated event to broker  4. Implement list-pending endpoint  5. Write integration tests | Lead | 8 |

## 3. Detailed Tasks & Schedule

| **Day** | **Tasks** | **Owner** |
| --- | --- | --- |
| Day 1 | - Kickoff: repo setup, install dependencies, configure environment files, set up message broker- Initialize frontend scaffolding: create Vite/React app, configure routing, install API client library, establish project structure- Define folder structure & modules (/auth, /profile, /jobs, /frontend) | Lead |
| Day 2–3 | **Account & Onboarding**: implement signup, login, verify controllers, connect to DB, queue email jobs, JWT middleware | Lead |
| Day 4–5 | **Profile Management**: define schema, build GET/PUT endpoints, add validation, write unit tests, generate Swagger docs | Lead |
| Day 6–8 | **Job Submission**: define schema, implement create & list endpoints, integrate event publishing, write integration tests | Lead |
| Day 9 | Wrap all routes behind feature-flag middleware; smoke-test each module in Postman | Lead |
| Day 10 | Buffer: refine error handling, write additional tests, update README with API specs | Lead |
| Day 11–12 | QA & bug fixes: manual API testing via Postman/Newman, address critical bugs | Lead |
| Day 13 | Sprint Review prep: compile API examples, Postman collection, CI pipeline status report | Lead |
| Day 14 | Sprint Review & Retrospective: demo APIs to stakeholders, perform retro, document action items | All team |

## 4. Definition of Done

* All backend endpoints implemented, tested, and feature-flagged
* Unit & integration tests passing
* Swagger/OpenAPI docs updated
* Postman collection ready
* CI pipeline green on this branch