Code Jam App

First presentable product:

- Team login
- Team writes code in embedded editor/IDE
- Submit button that sends info to backend
    - Backend stores submission and server time received
- Judge login
- Judging correct/incorrect (matching outputs)
- Scoreboard logic (does not need to be live data)
- Database storing mock questions (just a few)

Team	Goal by Feb 2	Deliverable	Comments
Networking	Get info regarding access to Frankie, SSH, ports, etc. Ensure Team and Judge sides can communicate via API	Resolution on use of Frankie, backup plan if not possible.	No physical deliverables, gameplan for communication between server and frontend/backend
Team Side	Communicate with the Security and Judge Side teams and choose which programming language you would like to use. In conjunction with the Judge Side team, define API calls needed. Determine if you want to code a custom IDE (consider 2 week timeframe for first working language) or embedding an IDE in the web page.	API calls with stubs/placeholder logic. API call to send data to the judge side. Determine selected programming language and environment for creating the app.	I recommend considering the Monaco development environment, which was made to be embedded. That combined with a language dropdown would constitute a working IDE for multiple programming languages. This would take the least amount of effort and give us time to focus on other aspects of the application.
Security	Login mechanism implementing access control between Judge and Team pages. Storage of login information.	No physical deliverable. Ensure that team login gets to the team page and admin login gets to the admin page.	Nothing fancy needed for authentication at the moment. Could just be random access code generation and verification for this first iteration.
Design	Minimum viable displays for: - Login (team/judge) - Team page - Judge page Including:	Basic UI that includes a preliminary layout for all needed displays: - Submit button that calls the API to send	Nothing needs to be crazy. This is the first iteration. Design can be simple for the scoreboard, buttons, etc.

	- Buttons (beside submit and scoring, do not need functionality) - Scoreboard initial design - IDE location - Display for problems	data - Dropdown that selects the programming language - Embedding of the IDE if a pre-made editor is selected	Backend teams will develop the API calls that you need for event handlers/action listeners etc.
Judge Side	Communicate with the Security and Team Side teams and choose which programming language you would like to use. Implementation of base scoring logic to include marking of test cases as correct or incorrect and other metrics. We need 1 corroborative list of API calls so that logic is the same across the entire backend logic.	Scoreboard logic API calls with stubs/placeholder logic. Communication with database to pull questions, submissions, and timestamps. API to get a timestamp when a submission is received.	Please make sure there is one consolidated list of API calls so that you, the Team side team, security, and design teams know what to call.
Database	A small database that can at least hold problems and submissions.	A database with some mock for problems, submissions, timestamps, etc. Minimal schema that supports scoreboard queries.	Do not need a complete, finished database for the first iteration. Mock data is perfect. Ensure communication with the Judge team to provide data to the backend.