Khel AI MVP

API Assignment Handbook

Student-wise Assignment 1-5, API Objectives, Expected Outputs, Submission Format, and Admin Evaluation Notes

Prepared for	Khel AI MVP API Session
Purpose	To explain exactly what each student has to build and how each answer should be submitted for review and evaluation.
Assignment structure	Assignment 1 and 2 are common APIs. Assignment 3, 4, and 5 are student-specific APIs.
Evaluation intent	The final pasted answer should let the evaluator see the question statement, the objective, the expected behavior, and the submitted code in one place.

Core rule: The data input should follow the existing Khel AI data model as-is. Any summary, ranking, classification, or analytics logic should be performed inside the API service or computation layer.

1. What this handbook is for

This PDF explains all five assignments for each student. It defines what each API is supposed to do, what kind of output is expected, and how the answer should be submitted so that an evaluator - human or automated - can judge whether the pasted code actually meets the stated objective.

2. Common rules for all students

- Assignment 1 is the first common API: Match Scoreboard API.
- Assignment 2 is the second common API: Innings Summary API.
- Assignment 3, 4, and 5 are the three student-specific APIs assigned in this handbook.
- Students should preserve the input shape of the original model relations. Do not invent simplified or pre-processed input structures unless explicitly documented as response-side transformation.
- All calculations, summaries, classifications, and derived labels should be implemented in service/helper logic, not hard-coded in the route itself.
- Submission must be text-first and reviewable: no screenshots and no videos.

3. Student-to-API mapping

Student	A1	A2	A3	A4	A5
Student 1	Match Scoreboard API	Innings Summary API	Batter Scorecard API	Top Batter API	Batting Form API
Student 2	Match Scoreboard API	Innings Summary API	Bowler Scorecard API	Top Bowler API	Bowling Form API
Student 3	Match Scoreboard API	Innings Summary API	Over Summary API	Recent Balls API	Momentum API
Student 4	Match Scoreboard API	Innings Summary API	Extras Summary API	Wicket Log API	Discipline Report API
Student 5	Match Scoreboard API	Innings Summary API	Match State API	Required Run Rate API	Win Probability Label API

Recommended answer ordering for submission: API Name -> Objective -> Expected behavior -> Code/files pasted as-is -> Render URL -> GitHub repo URL -> Admin handover documentation.

Student 1 Assignment Sheet

Student allocation: This student must complete all five assignments below. Assignment 1 and Assignment 2 are common. Assignment 3 to 5 are the student-specific APIs listed for this student.

Assignment 1: Match Scoreboard API

Motto: Return the live scoreboard for a match in a clean, frontend-ready JSON format.

What this API should do:

- Accept a match identifier and fetch the active or latest innings for that match.
- Return the current score, wickets, overs, run rate, batting team, bowling team, and key performers.
- Handle the case where a match exists but no innings has been created yet.
- Keep calculations in a service/helper layer rather than hard-coding them inside the route.

What we are expecting from this API:

A stable live-score endpoint that can be integrated on the scoring page, live stream page, and later exposed as a tool to an AI agent.

Suggested response/output fields:

match | **venue** | **innings_number** | **batting_team** | **bowling_team** | **score** | **overs** | **run_rate** | **top_batter** | **top_bowler** | **recent_balls**

Submission header format for this assignment:

Line 1	Name of the API - Match Scoreboard API
Line 2	Objective of the API - a one-paragraph statement of what this API is supposed to do.
After that	Paste the code or the relevant files exactly as they are.
Then include	Render URL, GitHub repo URL, and admin handover documentation.

Assignment 2: Innings Summary API

Motto: Expose a full innings-level statistical summary from the raw ball-event data.

What this API should do:

- Accept an innings identifier and return complete innings summary data.
- Compute totals from the saved ball events rather than relying on manual summary fields.
- Return batter summaries, bowler summaries, legal balls, overs, wickets, and recent balls.
- Follow the same input shape as the existing data model and move business logic into services.

What we are expecting from this API:

A reusable summary endpoint that gives the system one reliable innings-level analytics object.

Suggested response/output fields:

total_runs | **wickets** | **legal_balls** | **overs** | **run_rate** | **batters** | **bowlers** | **top_batter** | **top_bowler** | **recent_balls**

Submission header format for this assignment:

Line 1	Name of the API - Innings Summary API
Line 2	Objective of the API - a one-paragraph statement of what this API is supposed to do.
After that	Paste the code or the relevant files exactly as they are.
Then include	Render URL, GitHub repo URL, and admin handover documentation.

Assignment 3: Batter Scorecard API

Motto: Turn raw striker-wise ball data into a clean batting card.

What this API should do:

- Return one list containing batter name, runs, balls, fours, sixes, and strike rate.
- Count only legal deliveries for balls faced unless the business rule explicitly says otherwise.
- Rank or order the card cleanly for display.

What we are expecting from this API:

A full batting card that the admin can show directly on a scoreboard or commentary panel.

Suggested response/output fields:

name | **runs** | **balls** | **fours** | **sixes** | **strike_rate**

Submission header format for this assignment:

Line 1	Name of the API - Batter Scorecard API
Line 2	Objective of the API - a one-paragraph statement of what this API is supposed to do.
After that	Paste the code or the relevant files exactly as they are.
Then include	Render URL, GitHub repo URL, and admin handover documentation.

Assignment 4: Top Batter API

Motto: Identify the most effective batter in the innings using the saved event data.

What this API should do:

- Return the top batter using a clearly defined rule such as highest runs.
- Optionally include supporting context such as strike rate and boundaries.
- Handle empty innings safely.

What we are expecting from this API:

A focused endpoint that surfaces the lead batting performer without forcing the frontend to compute it.

Suggested response/output fields:

name | **runs** | **balls** | **fours** | **sixes** | **strike_rate**

Submission header format for this assignment:

Line 1	Name of the API - Top Batter API
Line 2	Objective of the API - a one-paragraph statement of what this API is supposed to do.
After that	Paste the code or the relevant files exactly as they are.
Then include	Render URL, GitHub repo URL, and admin handover documentation.

Assignment 5: Batting Form API

Motto: Show how one batter has been performing across innings or recent matches.

What this API should do:

- Accept a player identifier.
- Aggregate the player's batting events across available innings or recent appearances.
- Return summary indicators such as runs, balls, strike rate, and boundary count.

What we are expecting from this API:

A player-centric endpoint useful for pre-match analysis, overlays, and later agent explanations.

Suggested response/output fields:

player | **innings_count** | **runs** | **balls** | **strike_rate** | **boundaries**

Submission header format for this assignment:

Line 1	Name of the API - Batting Form API

Line 2	Objective of the API - a one-paragraph statement of what this API is supposed to do.
After that	Paste the code or the relevant files exactly as they are.
Then include	Render URL, GitHub repo URL, and admin handover documentation.

Student 2 Assignment Sheet

Student allocation: This student must complete all five assignments below. Assignment 1 and Assignment 2 are common. Assignment 3 to 5 are the student-specific APIs listed for this student.

Assignment 1: Match Scoreboard API

Motto: Return the live scoreboard for a match in a clean, frontend-ready JSON format.

What this API should do:

- Accept a match identifier and fetch the active or latest innings for that match.
- Return the current score, wickets, overs, run rate, batting team, bowling team, and key performers.
- Handle the case where a match exists but no innings has been created yet.
- Keep calculations in a service/helper layer rather than hard-coding them inside the route.

What we are expecting from this API:

A stable live-score endpoint that can be integrated on the scoring page, live stream page, and later exposed as a tool to an AI agent.

Suggested response/output fields:

match | **venue** | **innings_number** | **batting_team** | **bowling_team** | **score** | **overs** | **run_rate** | **top_batter** | **top_bowler** | **recent_balls**

Submission header format for this assignment:

Line 1	Name of the API - Match Scoreboard API
Line 2	Objective of the API - a one-paragraph statement of what this API is supposed to do.
After that	Paste the code or the relevant files exactly as they are.
Then include	Render URL, GitHub repo URL, and admin handover documentation.

Assignment 2: Innings Summary API

Motto: Expose a full innings-level statistical summary from the raw ball-event data.

What this API should do:

- Accept an innings identifier and return complete innings summary data.
- Compute totals from the saved ball events rather than relying on manual summary fields.
- Return batter summaries, bowler summaries, legal balls, overs, wickets, and recent balls.
- Follow the same input shape as the existing data model and move business logic into services.

What we are expecting from this API:

A reusable summary endpoint that gives the system one reliable innings-level analytics object.

Suggested response/output fields:

total_runs | **wickets** | **legal_balls** | **overs** | **run_rate** | **batters** | **bowlers** | **top_batter** | **top_bowler** | **recent_balls**

Submission header format for this assignment:

Line 1	Name of the API - Innings Summary API
Line 2	Objective of the API - a one-paragraph statement of what this API is supposed to do.
After that	Paste the code or the relevant files exactly as they are.
Then include	Render URL, GitHub repo URL, and admin handover documentation.

Assignment 3: Bowler Scorecard API

Motto: Convert bowling events into a clear bowling card.

What this API should do:

- Return bowler name, balls, overs, runs conceded, wickets, and economy.
- Use legal-delivery rules correctly when calculating overs and economy.
- Keep wicket credit rules correct for run out and retired out cases if your logic handles those separately.

What we are expecting from this API:

A reliable bowling card ready for frontend use.

Suggested response/output fields:

name | **balls** | **overs** | **runs_conceded** | **wickets** | **economy**

Submission header format for this assignment:

Line 1	Name of the API - Bowler Scorecard API
Line 2	Objective of the API - a one-paragraph statement of what this API is supposed to do.
After that	Paste the code or the relevant files exactly as they are.
Then include	Render URL, GitHub repo URL, and admin

	handover documentation.

Assignment 4: Top Bowler API

Motto: Surface the bowler making the biggest impact in the innings.

What this API should do:

- Choose the best bowler using a declared rule, such as wickets first and economy next.
- Return enough context for the frontend to display why that bowler is leading.
- Handle innings with no bowling data safely.

What we are expecting from this API:

A single-result impact endpoint that can power quick summary widgets.

Suggested response/output fields:

name | **overs** | **runs_conceded** | **wickets** | **economy**

Submission header format for this assignment:

Line 1	Name of the API - Top Bowler API
Line 2	Objective of the API - a one-paragraph statement of what this API is supposed to do.
After that	Paste the code or the relevant files exactly as they are.
Then include	Render URL, GitHub repo URL, and admin handover documentation.

Assignment 5: Bowling Form API

Motto: Track how one bowler has been performing across innings or matches.

What this API should do:

- Accept a player identifier.
- Aggregate bowling records across multiple innings or recent appearances.
- Return wickets, balls, overs, runs conceded, and economy.

What we are expecting from this API:

A player-centric bowling analytics endpoint.

Suggested response/output fields:

player | **innings_count** | **balls** | **overs** | **runs_conceded** | **wickets** | **economy**

Submission header format for this assignment:

Line 1	Name of the API - Bowling Form API
Line 2	Objective of the API - a one-paragraph statement of what this API is supposed to do.
After that	Paste the code or the relevant files exactly as they are.
Then include	Render URL, GitHub repo URL, and admin handover documentation.

Student 3 Assignment Sheet

Student allocation: This student must complete all five assignments below. Assignment 1 and Assignment 2 are common. Assignment 3 to 5 are the student-specific APIs listed for this student.

Assignment 1: Match Scoreboard API

Motto: Return the live scoreboard for a match in a clean, frontend-ready JSON format.

What this API should do:

- Accept a match identifier and fetch the active or latest innings for that match.
- Return the current score, wickets, overs, run rate, batting team, bowling team, and key performers.
- Handle the case where a match exists but no innings has been created yet.
- Keep calculations in a service/helper layer rather than hard-coding them inside the route.

What we are expecting from this API:

A stable live-score endpoint that can be integrated on the scoring page, live stream page, and later exposed as a tool to an AI agent.

Suggested response/output fields:

match | **venue** | **innings_number** | **batting_team** | **bowling_team** | **score** | **overs** | **run_rate** | **top_batter** | **top_bowler** | **recent_balls**

Submission header format for this assignment:

Line 1	Name of the API - Match Scoreboard API
Line 2	Objective of the API - a one-paragraph statement of what this API is supposed to do.
After that	Paste the code or the relevant files exactly as they are.
Then include	Render URL, GitHub repo URL, and admin handover documentation.

Assignment 2: Innings Summary API

Motto: Expose a full innings-level statistical summary from the raw ball-event data.

What this API should do:

- Accept an innings identifier and return complete innings summary data.
- Compute totals from the saved ball events rather than relying on manual summary fields.
- Return batter summaries, bowler summaries, legal balls, overs, wickets, and recent balls.
- Follow the same input shape as the existing data model and move business logic into services.

What we are expecting from this API:

A reusable summary endpoint that gives the system one reliable innings-level analytics object.

Suggested response/output fields:

total_runs | **wickets** | **legal_balls** | **overs** | **run_rate** | **batters** | **bowlers** | **top_batter** | **top_bowler** | **recent_balls**

Submission header format for this assignment:

Line 1	Name of the API - Innings Summary API
Line 2	Objective of the API - a one-paragraph statement of what this API is supposed to do.
After that	Paste the code or the relevant files exactly as they are.
Then include	Render URL, GitHub repo URL, and admin handover documentation.

Assignment 3: Over Summary API

Motto: Explain the innings one over at a time.

What this API should do:

- Group ball events over-wise.
- Return runs, wickets, extras, and ball labels for each over.
- Keep the output easy for timeline display.

What we are expecting from this API:

A scrolling or chart-ready over-by-over feed.

Suggested response/output fields:

over_number | **runs_in_over** | **wickets_in_over** | **extras_in_over** | **balls**

Submission header format for this assignment:

Line 1	Name of the API - Over Summary API
Line 2	Objective of the API - a one-paragraph statement of what this API is supposed to do.
After that	Paste the code or the relevant files exactly as they are.
Then include	Render URL, GitHub repo URL, and admin handover documentation.

Assignment 4: Recent Balls API

Motto: Return the last few events in a compact commentary-friendly structure.

What this API should do:

- Return the most recent six to twelve balls in correct order.
- Include over.ball, striker, bowler, runs, wicket flag, and a human-friendly label.
- Avoid expensive or unnecessary extra fields.

What we are expecting from this API:

A lightweight live-update endpoint for score widgets and commentary strips.

Suggested response/output fields:

over_ball | **striker** | **bowler** | **runs** | **wicket** | **label**

Submission header format for this assignment:

Line 1	Name of the API - Recent Balls API
Line 2	Objective of the API - a one-paragraph statement of what this API is supposed to do.
After that	Paste the code or the relevant files exactly as they are.
Then include	Render URL, GitHub repo URL, and admin handover documentation.

Assignment 5: Momentum API

Motto: Translate over-by-over events into an innings momentum picture.

What this API should do:

- Compute trend information over recent overs.
- Return a momentum label such as steady, attacking, recovery, or collapse using a declared rule.
- Show enough supporting numbers that the result is explainable.

What we are expecting from this API:

An interpretable trend endpoint, not a black box.

Suggested response/output fields:

recent_overs | **runs_pattern** | **wickets_pattern** | **momentum_label**

Submission header format for this assignment:

Line 1	Name of the API - Momentum API

Line 2	Objective of the API - a one-paragraph statement of what this API is supposed to do.
After that	Paste the code or the relevant files exactly as they are.
Then include	Render URL, GitHub repo URL, and admin handover documentation.

Student 4 Assignment Sheet

Student allocation: This student must complete all five assignments below. Assignment 1 and Assignment 2 are common. Assignment 3 to 5 are the student-specific APIs listed for this student.

Assignment 1: Match Scoreboard API

Motto: Return the live scoreboard for a match in a clean, frontend-ready JSON format.

What this API should do:

- Accept a match identifier and fetch the active or latest innings for that match.
- Return the current score, wickets, overs, run rate, batting team, bowling team, and key performers.
- Handle the case where a match exists but no innings has been created yet.
- Keep calculations in a service/helper layer rather than hard-coding them inside the route.

What we are expecting from this API:

A stable live-score endpoint that can be integrated on the scoring page, live stream page, and later exposed as a tool to an AI agent.

Suggested response/output fields:

match | **venue** | **innings_number** | **batting_team** | **bowling_team** | **score** | **overs** | **run_rate** | **top_batter** | **top_bowler** | **recent_balls**

Submission header format for this assignment:

Line 1	Name of the API - Match Scoreboard API
Line 2	Objective of the API - a one-paragraph statement of what this API is supposed to do.
After that	Paste the code or the relevant files exactly as they are.
Then include	Render URL, GitHub repo URL, and admin handover documentation.

Assignment 2: Innings Summary API

Motto: Expose a full innings-level statistical summary from the raw ball-event data.

What this API should do:

- Accept an innings identifier and return complete innings summary data.
- Compute totals from the saved ball events rather than relying on manual summary fields.
- Return batter summaries, bowler summaries, legal balls, overs, wickets, and recent balls.
- Follow the same input shape as the existing data model and move business logic into services.

What we are expecting from this API:
A reusable summary endpoint that gives the system one reliable innings-level analytics object.

Suggested response/output fields:
total_runs | **wickets** | **legal_balls** | **overs** | **run_rate** | **batters** | **bowlers** | **top_batter** | **top_bowler** | **recent_balls**

Submission header format for this assignment:

Line 1	Name of the API - Innings Summary API
Line 2	Objective of the API - a one-paragraph statement of what this API is supposed to do.
After that	Paste the code or the relevant files exactly as they are.
Then include	Render URL, GitHub repo URL, and admin handover documentation.

Assignment 3: Extras Summary API

Motto: Break down all extras cleanly from the innings data.

What this API should do:
- Count wides, no balls, byes, leg byes, penalties, and total extras.
- Use the existing extra_type and extras fields from the model.
- Return a clean breakdown that can be shown directly in notes or dashboards.

What we are expecting from this API:
A disciplined extras view for review and analysis.

Suggested response/output fields:
total_extras | **wides** | **no_balls** | **byes** | **leg_byes** | **penalties**

Submission header format for this assignment:

Line 1	Name of the API - Extras Summary API
Line 2	Objective of the API - a one-paragraph statement of what this API is supposed to do.
After that	Paste the code or the relevant files exactly as they are.
Then include	Render URL, GitHub repo URL, and admin handover documentation.

Assignment 4: Wicket Log API

Motto: Tell the wicket story of the innings in sequence.

What this API should do:

- Return each wicket event in innings order.
- Include over.ball, dismissed player, wicket type, bowler, fielder, and notes if available.
- Handle wicketless innings safely.

What we are expecting from this API:

A timeline endpoint for wickets, collapses, and turning points.

Suggested response/output fields:

over_ball | **dismissed_player** | **wicket_type** | **bowler** | **fielder_name** | **notes**

Submission header format for this assignment:

Line 1	Name of the API - Wicket Log API
Line 2	Objective of the API - a one-paragraph statement of what this API is supposed to do.
After that	Paste the code or the relevant files exactly as they are.
Then include	Render URL, GitHub repo URL, and admin handover documentation.

Assignment 5: Discipline Report API

Motto: Measure bowling discipline from wides, no-balls, and illegal deliveries.

What this API should do:

- Return counts of illegal deliveries and relevant discipline indicators.
- Highlight which bowlers contributed most to indiscipline if that is part of the design.
- Keep the rule for discipline score clearly documented.

What we are expecting from this API:

An analysis endpoint that reveals control issues without manual calculation on the frontend.

Suggested response/output fields:

illegal_deliveries | **wides** | **no_balls** | **discipline_score** | **worst_offender**

Submission header format for this assignment:

Line 1	Name of the API - Discipline Report API

Line 2	Objective of the API - a one-paragraph statement of what this API is supposed to do.
After that	Paste the code or the relevant files exactly as they are.
Then include	Render URL, GitHub repo URL, and admin handover documentation.

Student 5 Assignment Sheet

Student allocation: This student must complete all five assignments below. Assignment 1 and Assignment 2 are common. Assignment 3 to 5 are the student-specific APIs listed for this student.

Assignment 1: Match Scoreboard API

Motto: Return the live scoreboard for a match in a clean, frontend-ready JSON format.

What this API should do:

- Accept a match identifier and fetch the active or latest innings for that match.
- Return the current score, wickets, overs, run rate, batting team, bowling team, and key performers.
- Handle the case where a match exists but no innings has been created yet.
- Keep calculations in a service/helper layer rather than hard-coding them inside the route.

What we are expecting from this API:

A stable live-score endpoint that can be integrated on the scoring page, live stream page, and later exposed as a tool to an AI agent.

Suggested response/output fields:

match | **venue** | **innings_number** | **batting_team** | **bowling_team** | **score** | **overs** | **run_rate** | **top_batter** | **top_bowler** | **recent_balls**

Submission header format for this assignment:

Line 1	Name of the API - Match Scoreboard API
Line 2	Objective of the API - a one-paragraph statement of what this API is supposed to do.
After that	Paste the code or the relevant files exactly as they are.
Then include	Render URL, GitHub repo URL, and admin handover documentation.

Assignment 2: Innings Summary API

Motto: Expose a full innings-level statistical summary from the raw ball-event data.

What this API should do:

- Accept an innings identifier and return complete innings summary data.
- Compute totals from the saved ball events rather than relying on manual summary fields.
- Return batter summaries, bowler summaries, legal balls, overs, wickets, and recent balls.
- Follow the same input shape as the existing data model and move business logic into services.

What we are expecting from this API:

A reusable summary endpoint that gives the system one reliable innings-level analytics object.

Suggested response/output fields:

total_runs | **wickets** | **legal_balls** | **overs** | **run_rate** | **batters** | **bowlers** | **top_batter** | **top_bowler** | **recent_balls**

Submission header format for this assignment:

Line 1	Name of the API - Innings Summary API
Line 2	Objective of the API - a one-paragraph statement of what this API is supposed to do.
After that	Paste the code or the relevant files exactly as they are.
Then include	Render URL, GitHub repo URL, and admin handover documentation.

Assignment 3: Match State API

Motto: Return the current competitive state of the match.

What this API should do:

- Return current innings, score, overs, wickets, batting team, and bowling team.
- If chasing, include target context when available.
- Work as a summary endpoint that other features can build on.

What we are expecting from this API:

A core match-context API for live display and later agent use.

Suggested response/output fields:

match | **innings_number** | **score** | **overs** | **wickets** | **batting_team** | **bowling_team** | **target**

Submission header format for this assignment:

Line 1	Name of the API - Match State API
Line 2	Objective of the API - a one-paragraph statement of what this API is supposed to do.
After that	Paste the code or the relevant files exactly as they are.
Then include	Render URL, GitHub repo URL, and admin handover documentation.

Assignment 4: Required Run Rate API

Motto: Explain the chase equation clearly.

What this API should do:

- Use target, current score, and balls remaining to calculate the chase requirement.
- Return runs needed, balls remaining, and required run rate.
- Fail gracefully if the match is not currently a chase situation.

What we are expecting from this API:

A chase intelligence endpoint suitable for overlays and chatbot answers.

Suggested response/output fields:

target | **current_score** | **runs_needed** | **balls_remaining** | **required_run_rate**

Submission header format for this assignment:

Line 1	Name of the API - Required Run Rate API
Line 2	Objective of the API - a one-paragraph statement of what this API is supposed to do.
After that	Paste the code or the relevant files exactly as they are.
Then include	Render URL, GitHub repo URL, and admin handover documentation.

Assignment 5: Win Probability Label API

Motto: Provide a simple interpretable match-outlook label from the current chase situation.

What this API should do:

- Use a declared heuristic based on wickets in hand, overs left, current rate, and required rate.
- Return a label such as High Chance, Medium Chance, or Low Chance.
- Document the rule or logic clearly so the result can be defended.

What we are expecting from this API:

A simple explainable prediction label, not an unsupported magic number.

Suggested response/output fields:

runs_needed | **balls_remaining** | **required_run_rate** | **prediction**

Submission header format for this assignment:

Line 1	Name of the API - Win Probability Label API

Line 2	Objective of the API - a one-paragraph statement of what this API is supposed to do.
After that	Paste the code or the relevant files exactly as they are.
Then include	Render URL, GitHub repo URL, and admin handover documentation.

Submission template to be followed by every student

The submission should be structured so that an evaluator can read the question statement, understand the objective, inspect the code, and determine whether the implementation matches the claim.

Assignment X
API Name: <write exact API name here>
Objective: <write what the API is supposed to do>

Code / Files:
<paste main.py>
<paste schemas.py>
<paste services.py>
<paste any additional relevant file>

Render URL: <paste deployed URL>
GitHub Repo URL: <paste repo URL>

Admin Handover Documentation:
- Purpose
- Input schema
- Output schema
- Example request
- Example response
- Where this API can be used
- Possible validation or runtime errors

> **Why this structure matters:** When the answer is reviewed, the evaluator should be able to compare the objective with the actual code and decide whether the implementation really achieves the stated behavior.

What students should not submit

- No screenshots
- No video links
- No undocumented code dumps
- No missing repository or deployment link

Admin notes for evaluation

- The evaluator should first read the API Name and Objective line before looking at the code.
- The evaluator should verify whether the implementation accepts the appropriate input and produces the declared output.
- If the API claims analytics or predictions, the underlying rule or service logic should be identifiable in the code.

- The evaluator should check whether the Render URL works and whether the GitHub repository matches the submitted code.
- The handover documentation should be usable by another developer or admin without needing verbal explanation from the student.