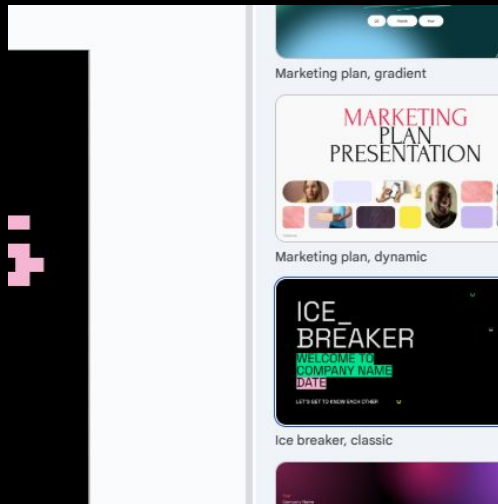
GUESS STREET

ITERATION 1

10/30/2025

BRAXTON / RYAN / JD / ELAINE

NOTE TO TEAM:
If you want to continue to use the slide template, It's called "Ice breaker, classic"



PROJECT SUMMARY:

Simple FastAPI service + local CLI that accepts user predictions for an asset's:

- target price range (min/max)
- target date

Stores them in Postgres, and exposes endpoints to create, list and fetch predictions.

CLIENT: Erman Pattuk:

- Software Engineer at Google (5 yrs)
- Software Engineer at LinkedIn (5 yrs)

ITERATION 1 FEATURES

Create Predictions

POST `/predictions` Create Prediction

Parameters | Try it out | Reset

No parameters

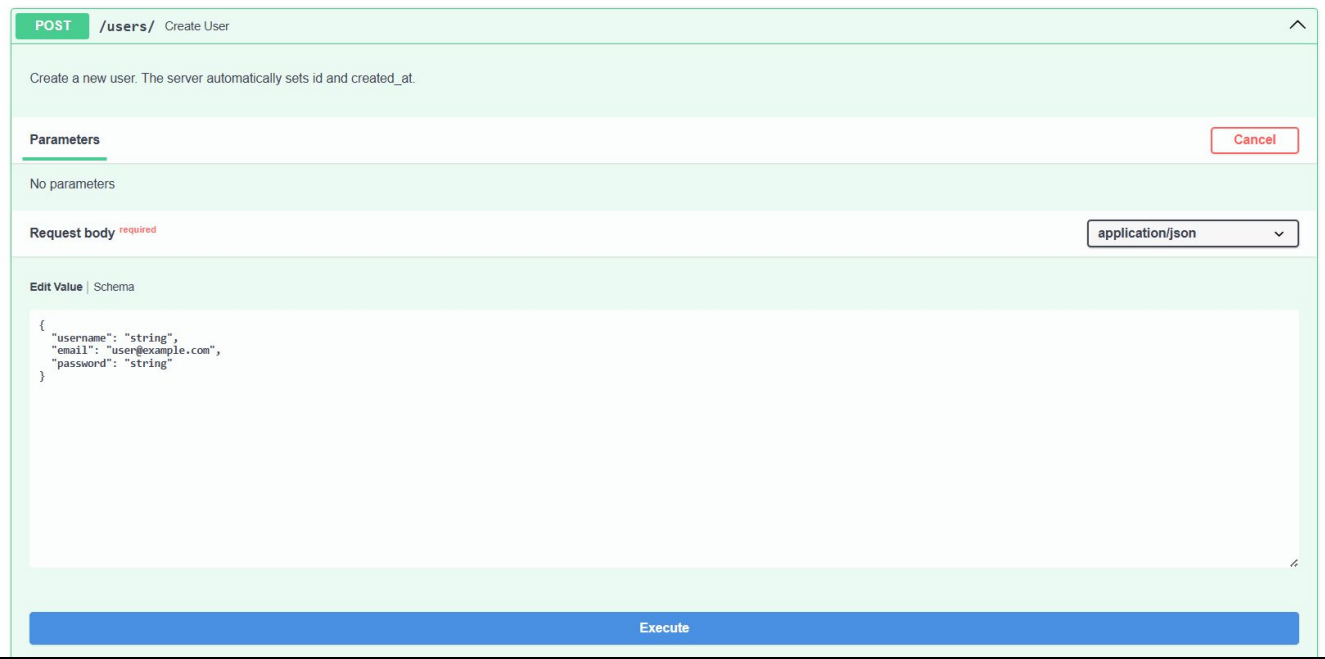
Request body required — application/json

Example Value | Schema


```
{
  "id": "3fa85f64-5717-4562-b3fc-2c963f66afa6",
  "asset": "string",
  "predictor": "string",
  "date_made": "2025-10-24T14:55:48.271Z",
  "min_value": 0,
  "max_value": 0,
  "target_date": "2025-10-24T14:55:48.271Z"
}
```

ITERATION 1 FEATURES

Create Users



ITERATION 1 FEATURES

Predictions are automatically stored in a Google Cloud Database

Responses

Curl

```
curl -X 'POST' \
  'http://127.0.0.1:8000/predictions' \
  -H 'accept: application/json' \
  -H 'Content-Type: application/json' \
  -d '{
  "id": "3fa85f64-5717-4562-b3fc-2c963f66afa6",
  "asset": "BTC",
  "predictor": "Ryan",
  "date_made": "2025-10-24T14:55:48.271Z",
  "min_value": 400,
  "max_value": 1000,
  "target_date": "2025-10-25T14:55:48.271Z"
}'
```

Request URL

```
http://127.0.0.1:8000/predictions
```

Server response

Code	Details
201	Response body

```
{
  "id": "55fa28d0-3238-4806-a791-2d7637c33a3a",
  "asset": "BTC",
  "predictor": "Ryan",
  "date_made": "2025-10-24T14:57:06.163916",
  "min_value": 400,
  "max_value": 1000,
  "target_date": "2025-10-25T10:55:48.271000"
}
```

Download

Response headers

```
content-length: 199
content-type: application/json
date: Fri,24 Oct 2025 14:57:05 GMT
server: uvicorn
```

ITERATION 1 FEATURES

Access all Predictions

GET /predictions List Predictions

Parameters

Cancel

No parameters

Execute

Clear

Responses

Curl

```
curl -X 'GET' \
  'http://127.0.0.1:8000/predictions' \
  -H 'accept: application/json'
```

Request URL

```
http://127.0.0.1:8000/predictions
```

Server response

Code Details

ITERATION 1 FEATURES

Access Predictions made by certain Users

GET /predictions/{user_id} List Predictions By User

Parameters

Try it out

Name	Description
user_id * required string (path)	user_id

Responses

Code	Description	Links
200	Successful Response Media type application/json Controls Accept header. Example Value | Schema	No links


```
[
  {
    "id": "3fa85f64-5717-4562-b3fc-2c963f66afa6",
    "asset": "string",
    "user_id": "string",
    "date_made": "2025-10-30T01:37:59.320Z",
    "min_value": 0,
    "max_value": 0,
    "target_date": "2025-10-30T01:37:59.320Z"
  }
]
```

ITERATION 1 FEATURES

GET /predictions/{prediction_id} Get Prediction

Parameters

Cancel

Name	Description
prediction_id * required string($uuid) (path)	prediction_id

Execute

Responses

Code	Description	Links
200	Successful Response Media type application/json Controls Accept header. Example Value | Schema	No links


```
{
  "id": "3fa85f64-5717-4562-b3fc-2c963f66afa6",
  "asset": "string",
  "user_id": "string",
  "date_made": "2025-10-30T01:36:56.026Z",
  "min_value": 0,
  "max_value": 0,
  "target_date": "2025-10-30T01:36:56.026Z"
```

(link if needed)

DEMO

CLIENT FEEDBACK

Satisfied with our direction:

“Sounds good.”

SECURITY

Poor security can lead to vulnerabilities.

He is responsible for anything that happens to the software

USER INPUT

User input should be minimized to the absolute best of our ability.

Any input could be malicious.

SERVER-SIDE VERIFICATION

The server should provide the database with attributes like timestamps and UUIDs.

Users should not be able to input into these fields.

WHAT WILL CHANGE?

Client feedback we (Ryan) changed immediately:

Server now generates UUIDs and timestamps.

Users can no longer input to these fields.

Client feedback we plan to change:

Implementing excessive input validation.

Iteration 2 features

PLANNED ITERATION 2 FEATURES

- Finish user Class and Tests (w/ Database)
- Get Authentication up and running
- Connect Users and Predictions
- Resolve Predictions with Market Outcomes

Retrospective

What does each team member think about this iteration?

Retrospective

What kind of properties of quality software did you sacrifice for functional software?

Security - users have too much access to the input values.

Usability - users must currently navigate an API to use.

PLAN TO APPROACH ITERATION 2

- Meetings similar to the end of the First Iteration
  - In Person
  - Discuss and divide work accordingly
- Work together on tasks likely to be difficult
  - Authentication
- Start earlier in general
  - Early unused time for Iteration 1

Interesting Slide

QUESTIONS?