Flask Findings: Good for building scalable web applications

- Good backend development
- Easy frontend integration
- Robust database support

How Flask Works:

- HTTP requests are routed to python functions
- Functions process requests and return responses, typically in the form of HTML, JSON, or other data formats

Initialize Flask App:

- Define a route (/) that maps to function returning response

```
from flask import Flask

app = Flask(__name__)

@app.route('/')
def home():
    return "Hello, Flask!"

if __name__ == '__main__':
    app.run(debug=True)
```

Backend:

- Handles data processing, authentication, API requests, among others
- Typically consists of
    - API endpoints
    - Database
    - Frontend using the backend API

```
from flask import Flask, jsonify

app = Flask(__name__)

@app.route('/api/data')
def get_data():
    return jsonify({"message": "This is some data!"})
```

```
if __name__ == '__main__':
    app.run(debug=True)
```

OR

```
from flask import Flask, jsonify

app = Flask(__name__)

@app.route('/api/users', methods=['GET'])
def get_users():
    users = [{"id": 1, "name": "Alice"}, {"id": 2, "name": "Bob"}]
    return jsonify(users)

if __name__ == '__main__':
    app.run(debug=True)
```

OR

```
from flask import Flask, request, jsonify

app = Flask(__name__)

# Example storage (use a database in production)
brackets = {}

@app.route('/api/bracket', methods=['POST'])
def submit_bracket():
    data = request.json
    user_id = data.get("user_id")
    brackets[user_id] = data.get("bracket")
    return jsonify({"message": "Bracket submitted!"})

@app.route('/api/bracket/<user_id>', methods=['GET'])
def get_bracket(user_id):
    return jsonify(brackets.get(user_id, {"message": "Bracket not found"}))

if __name__ == '__main__':
    app.run(debug=True)
```

Integrating with Frontend:

- Develop API endpoints in Flask
- Make API calls from the frontend to retrieve and display data

```
fetch("http://127.0.0.1:5000/api/data")
  .then(response => response.json())
  .then(data => console.log(data));
```

Using Flask's flask-cors package, we can enable CORS (Cross-Origin Resource Sharing)

```
from flask_cors import CORS
app = Flask(__name__)
CORS(app)
```

Flask with Databases (SQLAlchemy):

- Initialize database
- Define User model

```
from flask import Flask
from flask_sqlalchemy import SQLAlchemy

app = Flask(__name__)
app.config['SQLALCHEMY_DATABASE_URI'] = 'sqlite:///database.db'
db = SQLAlchemy(app)

class User(db.Model):
  id = db.Column(db.Integer, primary_key=True)
  name = db.Column(db.String(100), nullable=False)

db.create_all()
```

Basic User Login in Flask:

```
from flask import Flask, redirect, url_for
from flask_login import LoginManager, UserMixin, login_user, login_required

app = Flask(__name__)
app.secret_key = 'secret'
login_manager = LoginManager(app)
```

```
class User(UserMixin):
    def __init__(self, id):
        self.id = id

@login_manager.user_loader
def load_user(user_id):
    return User(user_id)

@app.route('/login')
def login():
    user = User(id=1)
    login_user(user)
    return redirect(url_for('protected'))

@app.route('/protected')
@login_required
def protected():
    return "You are logged in!"
```