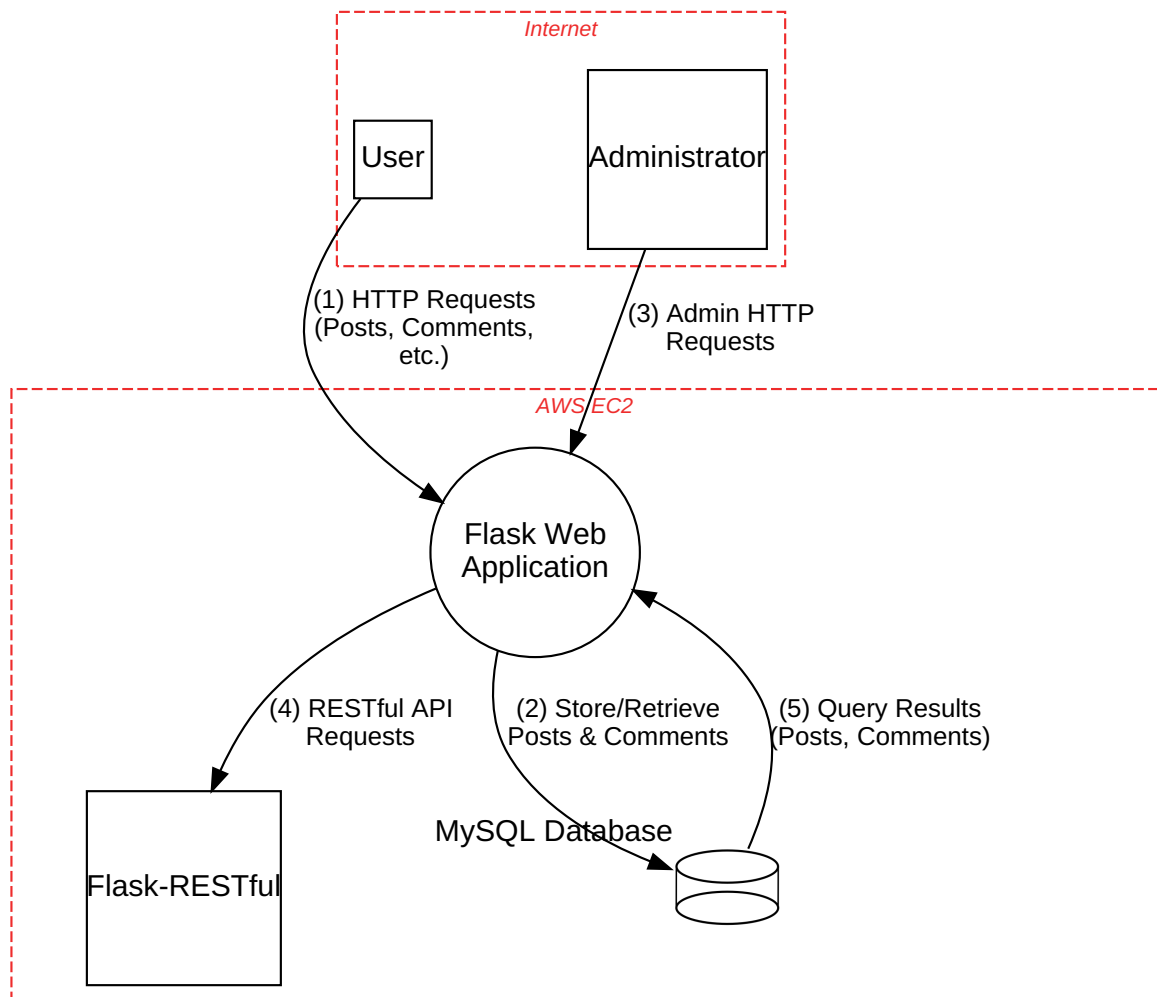




This is an auto-generated Data Flow Diagram, assembled by GPT-4 Threat Modeling Agents. The system reviews the specified application architecture, and generates a data flow diagram using pytm and GraphViz. The result may still contain errors.



Appendix

Original Prompt and Inputted App Architecture

Create a data flow diagram for the following app architecture: The application architecture is a web application with a database. The web application is written in Python and uses the Flask framework. The database is a MySQL database. The web application is hosted on AWS EC2. The web application is a simple blog application that allows users to create posts and comment on posts. The web application uses a MySQL database to store the posts and comments. The web application uses the Flask framework to handle requests and responses. The web application uses the Jinja2 templating engine to render HTML templates. The web application uses the WTForms library to handle forms. The web application uses the Flask-Login library to handle user authentication. The web application uses the Flask-WTF library to handle forms. The web application uses the Flask-Bootstrap library to handle forms. The web application uses the Flask-Admin library to handle forms. The web application uses the Flask-RESTful library to handle forms.

Usage Costs

Total Cost: \$0.03918 USD

Input Tokens Cost: \$0.0 USD

Output Tokens Cost: \$0.03918 USD

Conversation Log

| STARTING CONVERSATION: Create a data flow diagram for the following app architecture: The application architecture is a web application with a database. The web application is written in Python and uses the Flask framework. The database is a MySQL database. The web application is hosted on AWS EC2. The web application is a simple blog application that allows users to create posts and comment on posts. The web application uses a MySQL database to store the posts and comments. The web application uses the Flask framework to handle requests and responses. The web application uses the Jinja2 templating engine to render HTML templates. The web application uses the WTForms library to handle forms. The web application uses the Flask-Login library to handle user authentication. The web application uses the Flask-WTF library to handle forms. The web application uses the Flask-Bootstrap library to handle forms. The web application uses the Flask-Admin library to handle forms. The web application uses the Flask-RESTful library to handle forms. DESCRIPTION_END | user_proxy (to chatbot): | Create a data flow diagram for the following app architecture: The application architecture is a web application with a database. The web application is written in Python and uses the Flask framework. The database is a MySQL database. The web application is hosted on AWS EC2. The web application is a simple blog application that allows users to create posts and comment on posts. The web application uses a MySQL database to store the posts and comments. The web application uses the Flask framework to handle requests and responses. The web application uses the Jinja2 templating engine to render HTML templates. The web application uses the WTForms library to handle forms. The web application uses the Flask-Login library to handle user authentication. The web application uses the Flask-WTF library to handle forms. The web application uses the Flask-Bootstrap library to handle forms. The

```

web application uses the Flask-Admin library to handle forms. The web application uses the Flask-RESTful
library to handle forms.DESCRPTION_END | | ----- |
chatbot (to user_proxy): | | tm = TM("Blog Web Application Data Flow") | tm.description = "Data Flow Diagram
for a simple blog application hosted on AWS EC2 using Flask and MySQL" | tm.isOrdered = True |
tm.mergeResponses = True | | # Boundaries | internet_boundary = Boundary("Internet") | aws_ec2_boundary
= Boundary("AWS EC2") | | # Actors | user = Actor("User") | user.inBoundary = internet_boundary | |
administrator = Actor("Administrator") | administrator.inBoundary = internet_boundary | | # External Entities |
flask_restful = ExternalEntity("Flask-RESTful") | flask_restful.inBoundary = aws_ec2_boundary | | #
Processes | flask_app = Process("Flask Web Application") | flask_app.inBoundary = aws_ec2_boundary | | #
Datastores | mysql_db = Datastore("MySQL Database") | mysql_db.inBoundary = aws_ec2_boundary | | #
Data definitions | post_data = Data("Post Data") | comment_data = Data("Comment Data") | credentials =
Data("Credentials") | | # Dataflows | user_to_flask_app = Dataflow(user, flask_app, "HTTP Requests (Posts,
Comments, etc.)") | flask_app_to_mysql_db = Dataflow(flask_app, mysql_db, "Store/Retrieve Posts &
Comments") | admin_to_flask_app = Dataflow(administrator, flask_app, "Admin HTTP Requests") |
flask_app_to_restful_api = Dataflow(flask_app, flask_restful, "RESTful API Requests") |
mysql_db_to_flask_app = Dataflow(mysql_db, flask_app, "Query Results (Posts, Comments)") | | # Output
the data flow diagram | functions.python(cell=TM.to_json()) | * Suggested function Call: python |
Arguments: | {"cell": "tm = TM("Blog Web Application Data Flow")\n\n# Boundaries\ninternet_boundary = Boundary("Internet")\n\n# Actors\nuser = Actor("User")\nuser.inBoundary = internet_boundary\n\n# External Entities\nflask_restful = ExternalEntity("Flask-RESTful")\nflask_restful.inBoundary = aws_ec2_boundary\n\n# Processes\nflask_app = Process("Flask Web Application")\nflask_app.inBoundary = aws_ec2_boundary\n\n# Datastores\nmysql_db = Datastore("MySQL Database")\nmysql_db.inBoundary = aws_ec2_boundary\n\n# Data definitions\npost_data = Data("Post Data")\ncomment_data = Data("Comment Data")\ncredentials = Data("Credentials")\n\n# Dataflows\nuser_to_flask_app = Dataflow(user, flask_app, "HTTP Requests (Posts, Comments, etc.)")\nflask_app_to_mysql_db = Dataflow(flask_app, mysql_db, "Store/Retrieve Posts & Comments")\nadmin_to_flask_app = Dataflow(administrator, flask_app, "Admin HTTP Requests")\nflask_app_to_restful_api = Dataflow(flask_app, flask_restful, "RESTful API Requests")\nmysql_db_to_flask_app = Dataflow(mysql_db, flask_app, "Query Results (Posts, Comments)")\n"} |
***** | | ----- | | >>>>>>> EXECUTING FUNCTION
python... |

```