Case Study #3

GROCECOLL

GROCECOLL

- Grocery collection service
- Allows customers to create shopping lists that get collected and delivered by GroceColl's employees
- Available world-wide

GROCECOLL

- Employees have dedicated tablets displaying the list
- We need to design the collection side of the system
  - The customer side is already developed

Requirements

Functional

What the system should do

1. Web Based
2. Tablets receive list to be collected
3. Employees can mark items as collected or unavailable
4. When collection is done, the list should be transferred to payment engine
5. Offline support is a must

Non-Functional

What the system should deal with

NFR - What We Ask

1. “How many expected concurrent users?”	200
2. “How many lists will be processed per day?”	10,000
3. “What is the average size of a shopping list?”	500KB

NFR - What We Ask

4. “Do we need offline support?”	Yes!
5. “What is the desired SLA?”	Highest Possible
6. “How do lists arrive to the system?”	Queue

Data Volume

- 1 List = 500KB
- 10,000 lists / day = 5GB / day

=> ~2TB / year

Requirements

Functional

What the system should do

1. Web Based
2. Tablets receive list to be collected
3. Employees can mark items as collected or unavailable
4. When collection is done, the list should be transferred to payment engine
5. Offline support is a must

Non-Functional

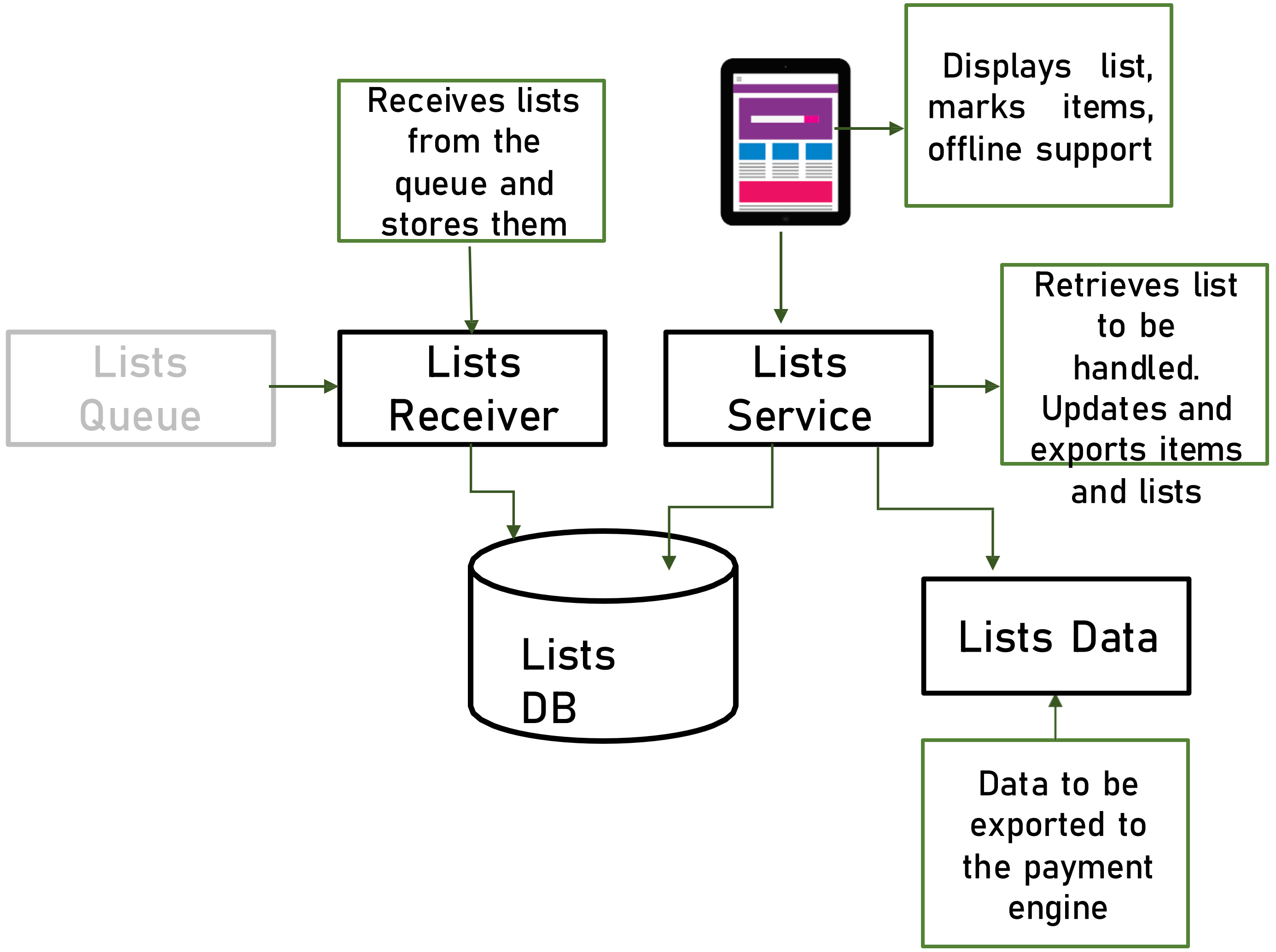
What the system should deal with

1. 200 Concurrent users
2. 10,000 lists/day
3. Yearly volume: 2TB
4. High SLA
5. Offline support

Components

Based on requirements:

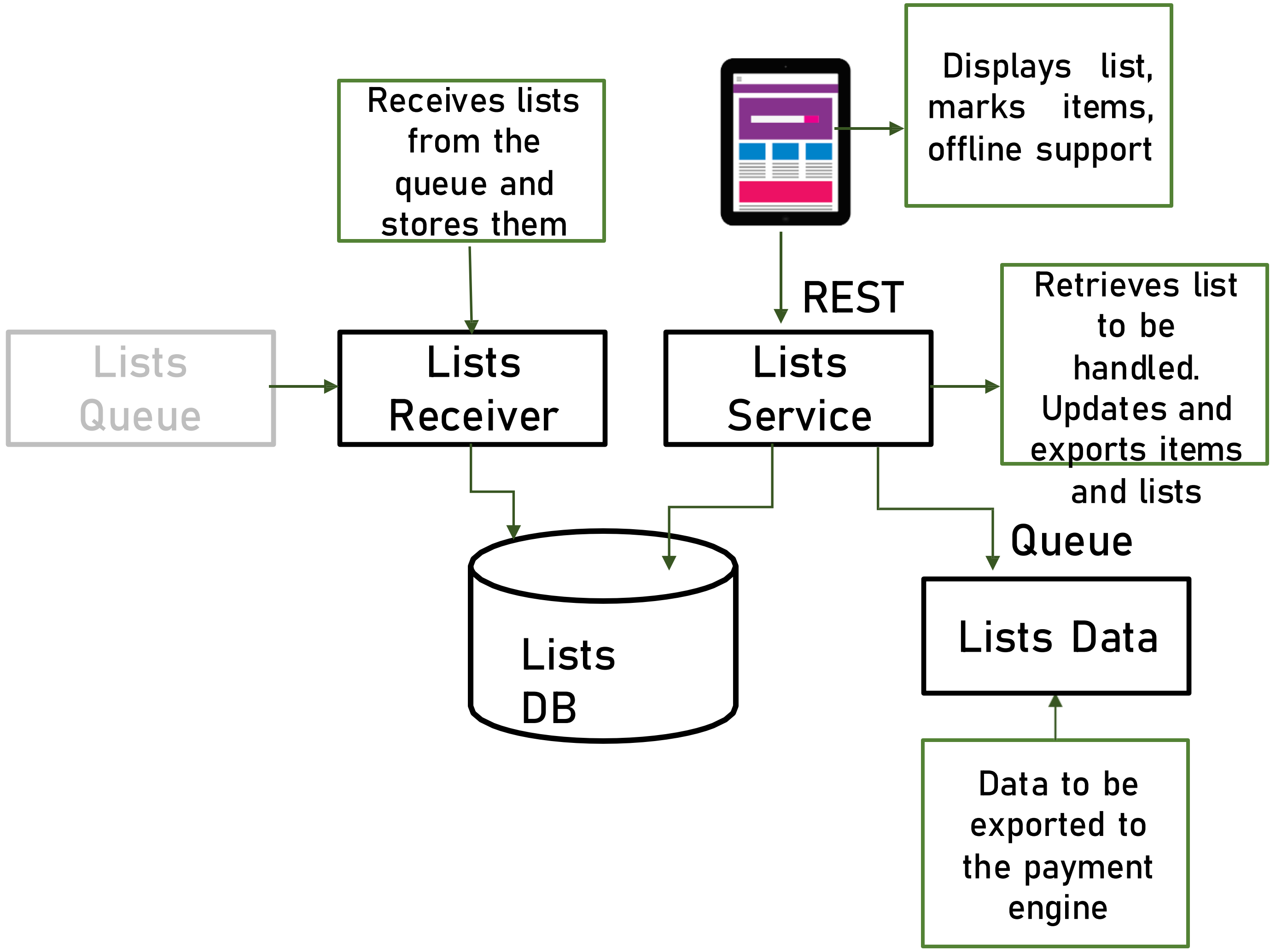
1. Employees have tablets
2. Offline support
3. Retrieve lists
4. Mark Items
5. Export list to payment engine


Receives lists from the queue and stores them
Displays list, marks items, offline support
Retrieves list to be handled. Updates and exports items and lists
Lists Queue
Lists Receiver
Lists Service
Lists DB
Lists Data
Data to be exported to the payment engine

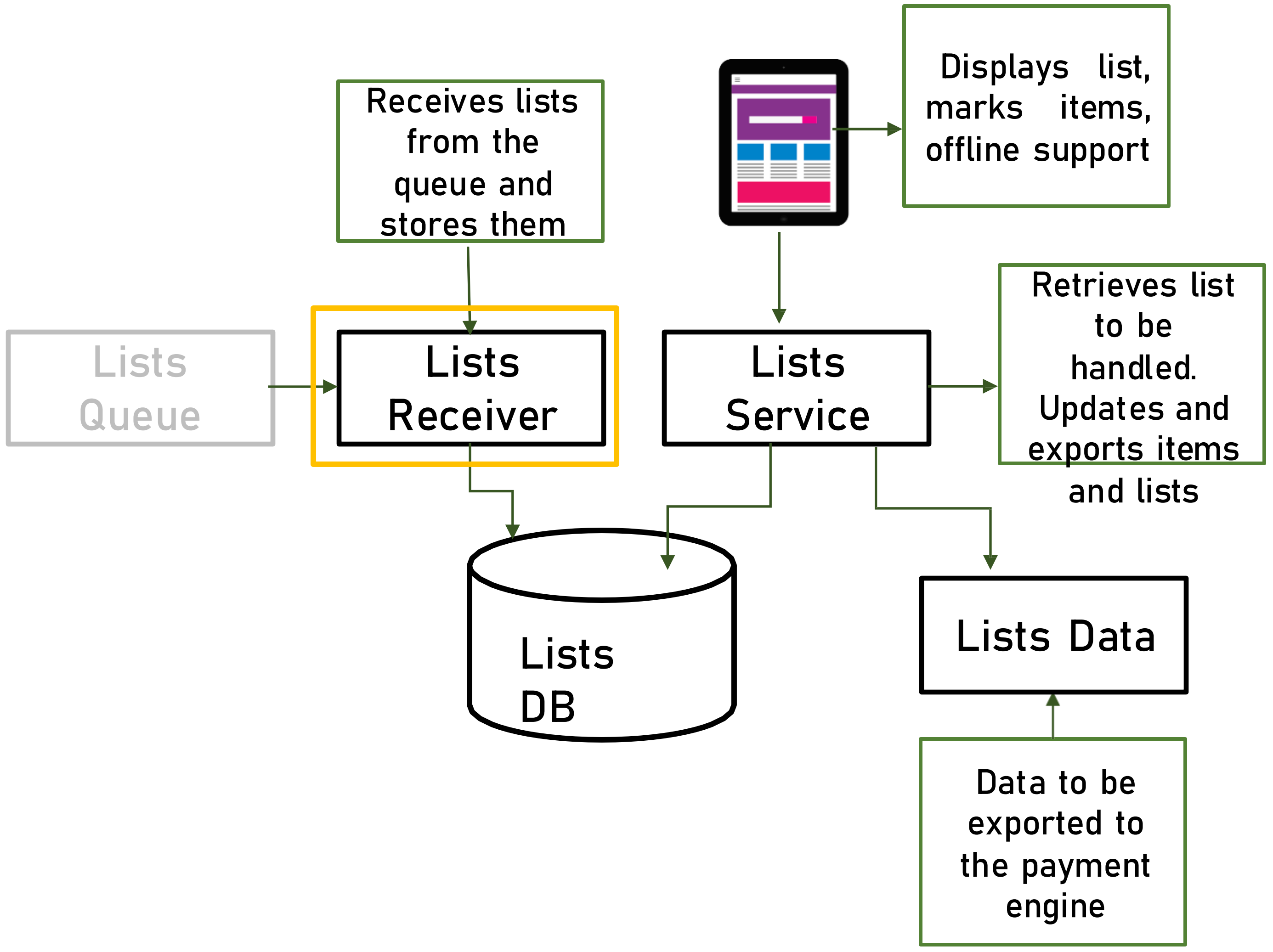
Messaging

Based on requirements:

1. Employees have tablets
2. Offline support
3. Retrieve lists
4. Mark Items
5. Export list to payment engine


Receives lists from the queue and stores them
Displays list, marks items, offline support
REST
Lists Queue
Lists Receiver
Lists Service
Retrieves list to be handled. Updates and exports items and lists
Queue
Lists DB
Lists Data
Data to be exported to the payment engine

Components


Receives lists from the queue and stores them
Displays list, marks items, offline support
Lists Queue
Lists Receiver
Lists Service
Retrieves list to be handled. Updates and exports items and lists
Lists DB
Lists Data
Data to be exported to the payment engine

Lists Receiver

What it does:

- Receives shopping lists to be handled from queue

- Stores the lists in the datastore

Application Type

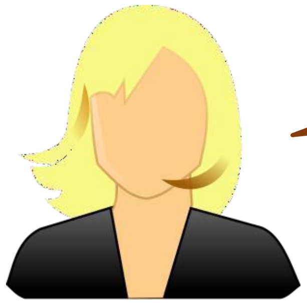
• Web App & Web API	✗
• Mobile App	✗
• Console	✓
• Service	✓
• Desktop App	✗

Technology Stack

Considerations:

- Should be able to connect to queue
- Not much else…

Technology Stack

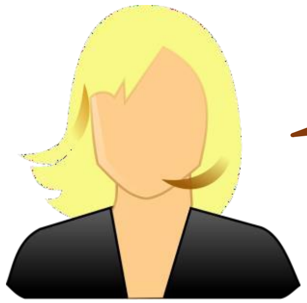


We’re basically a Java shop, and our database of choice is MySQL.

Java is a perfect fit for this task, so we’ll

go with it.

Technology Stack



We’re basically a Java shop, and our database of choice is MySQL.

What about database?

- Our data is relational, and MySQL is a relational DB
- Expected volume is 2TB/Year which is a lot
  - But can utilize partitioning
  - So…

Technology Stack

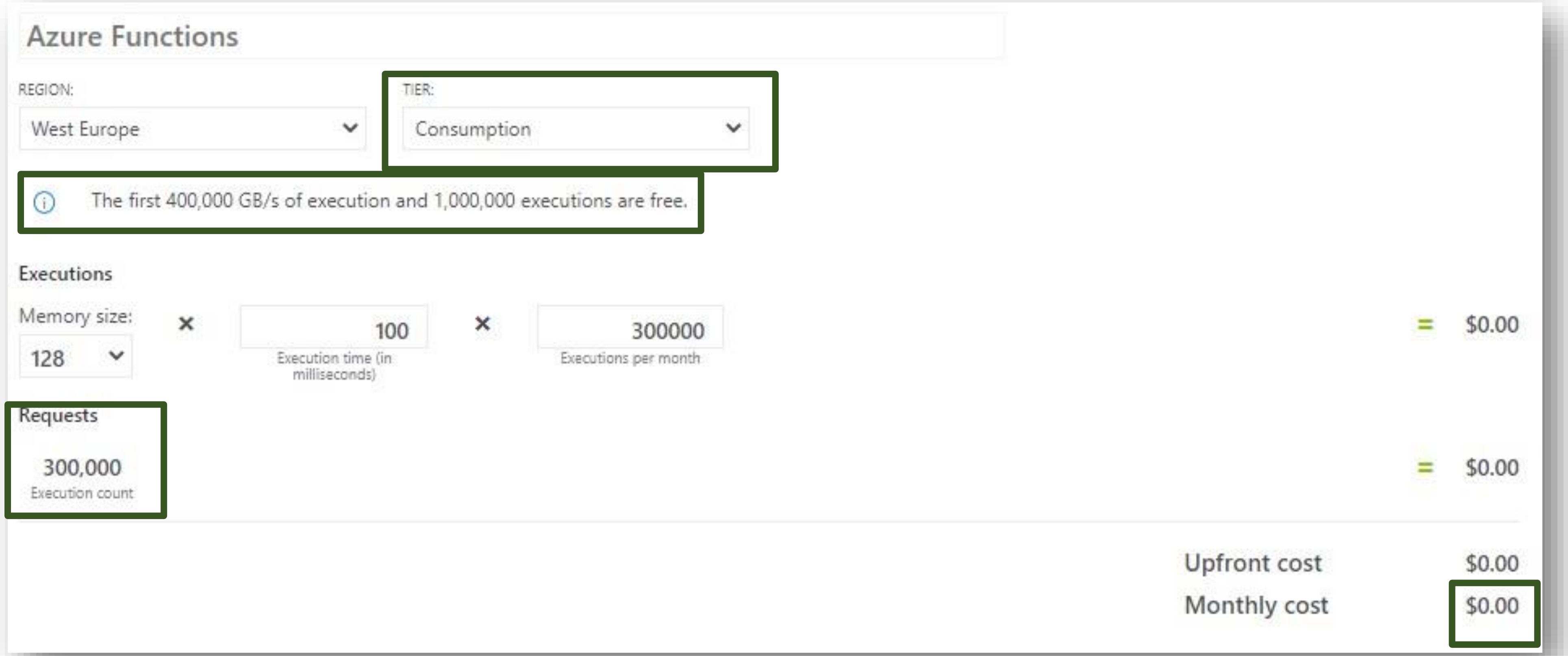
Java™

MySQL®

Receiver Code

Function App

- Designed for lightweight operations
- Great, built-in integration with many queue implementations
- Cost effective
- Autoscaling



Azure Functions

REGION: West Europe

TIER: Consumption

The first 400,000 GB/s of execution and 1,000,000 executions are free.

Executions

Memory size: 128 × 100 Execution time (in milliseconds) × 300000 Executions per month = $0.00

Requests

300,000 Execution count = $0.00

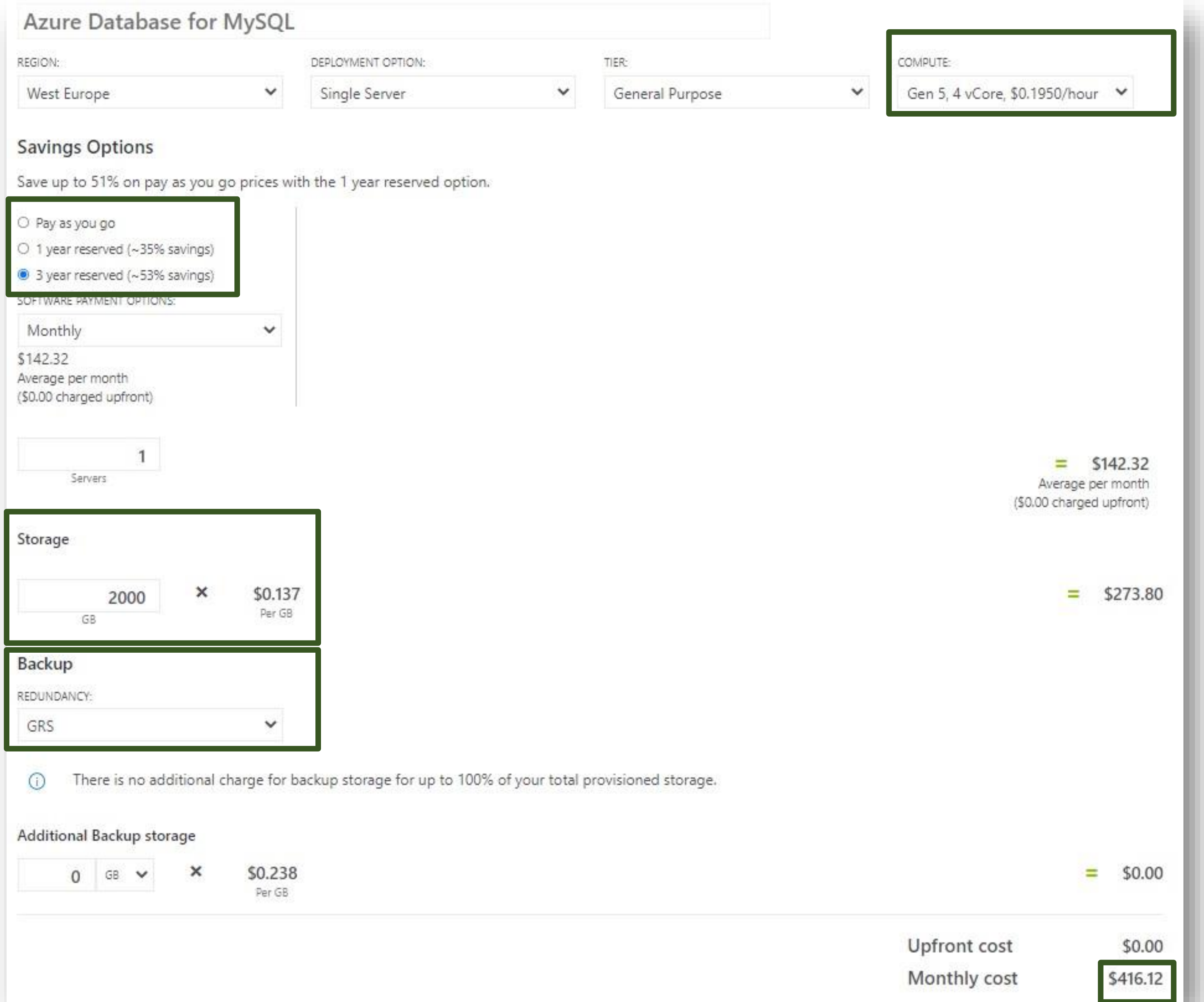
Upfront cost $0.00

Monthly cost $0.00

Receiver Database

Azure MySQL

- Fully managed MySQL in the cloud
- Automatic backup
- Scale up & down as needed


Azure Database for MySQL

REGION:	DEPLOYMENT OPTION:	TIER:	COMPUTE:
West Europe	Single Server	General Purpose	Gen 5, 4 vCore, $0.1950/hour

Savings Options

Save up to 51% on pay as you go prices with the 1 year reserved option.

- Pay as you go
- 1 year reserved (~35% savings)
- [✓ 3 year reserved (~53% savings)]

SOFTWARE PAYMENT OPTIONS: Monthly

$142.32
Average per month
($0.00 charged upfront)

1 Servers = $142.32 Average per month ($0.00 charged upfront)

Storage

2000 GB × $0.137 Per GB = $273.80

Backup

REDUNDANCY: GRS

There is no additional charge for backup storage for up to 100% of your total provisioned storage.

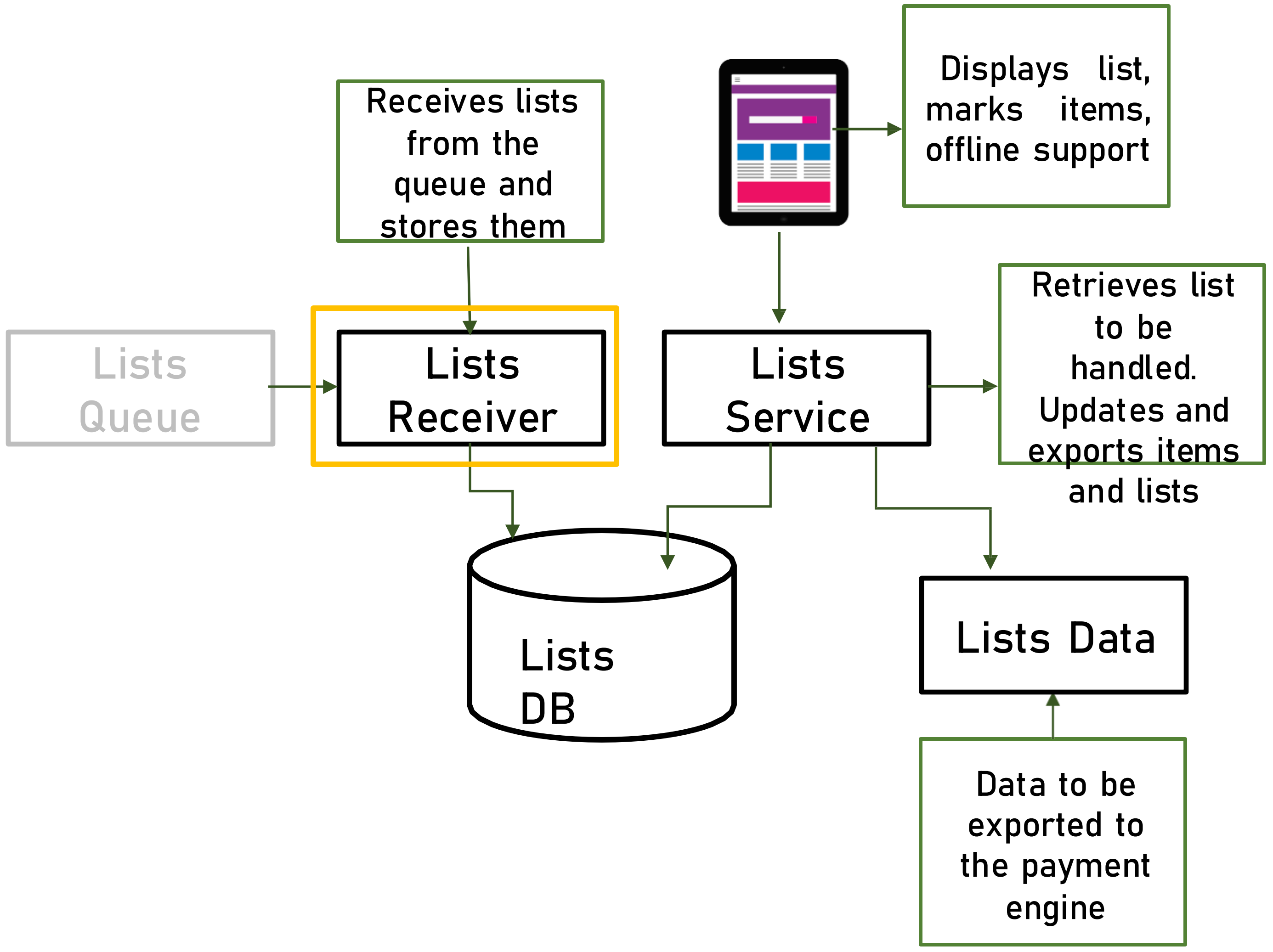
Additional Backup storage

0 GB × $0.238 Per GB = $0.00

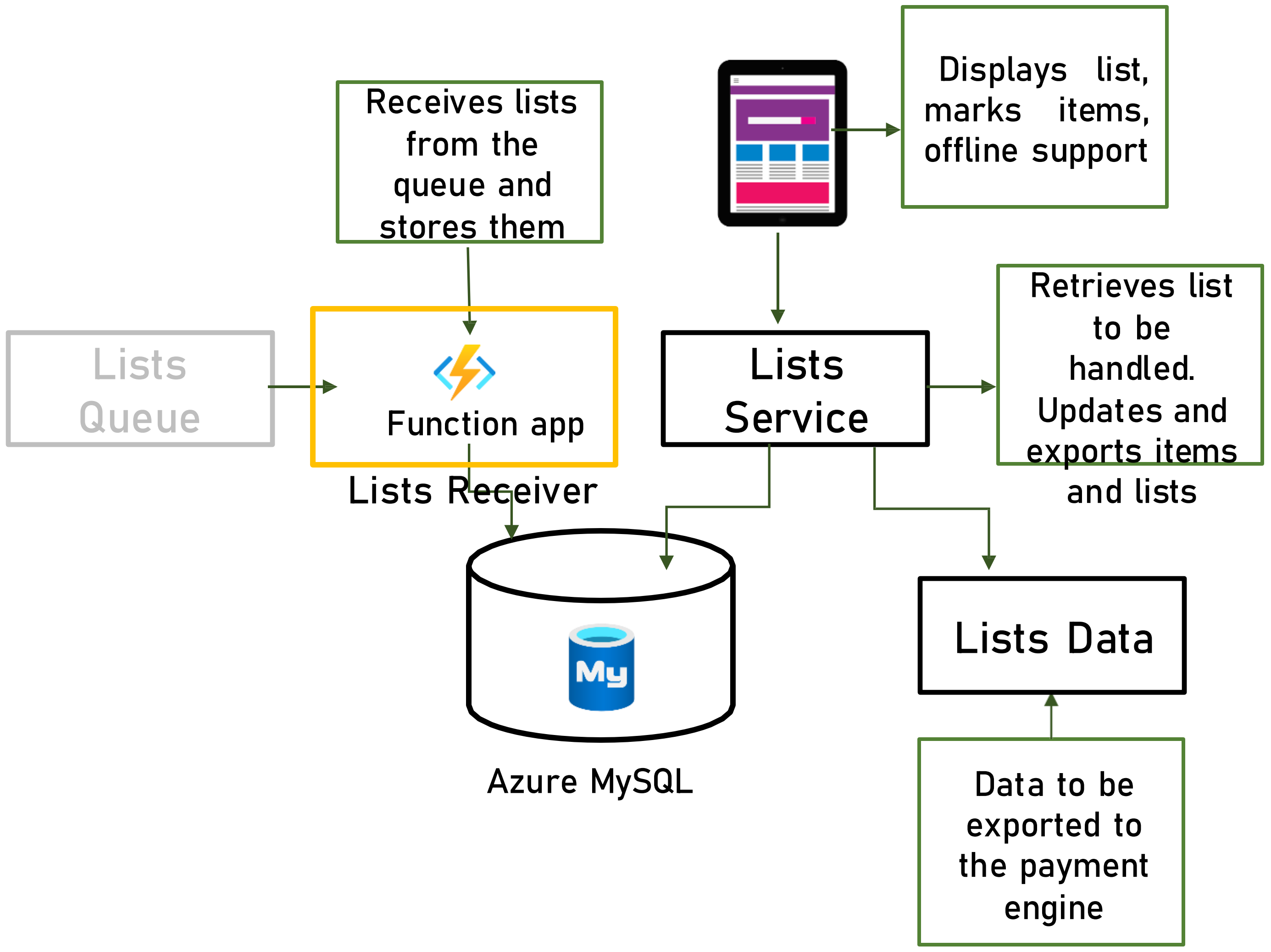
Upfront cost $0.00

Monthly cost $416.12

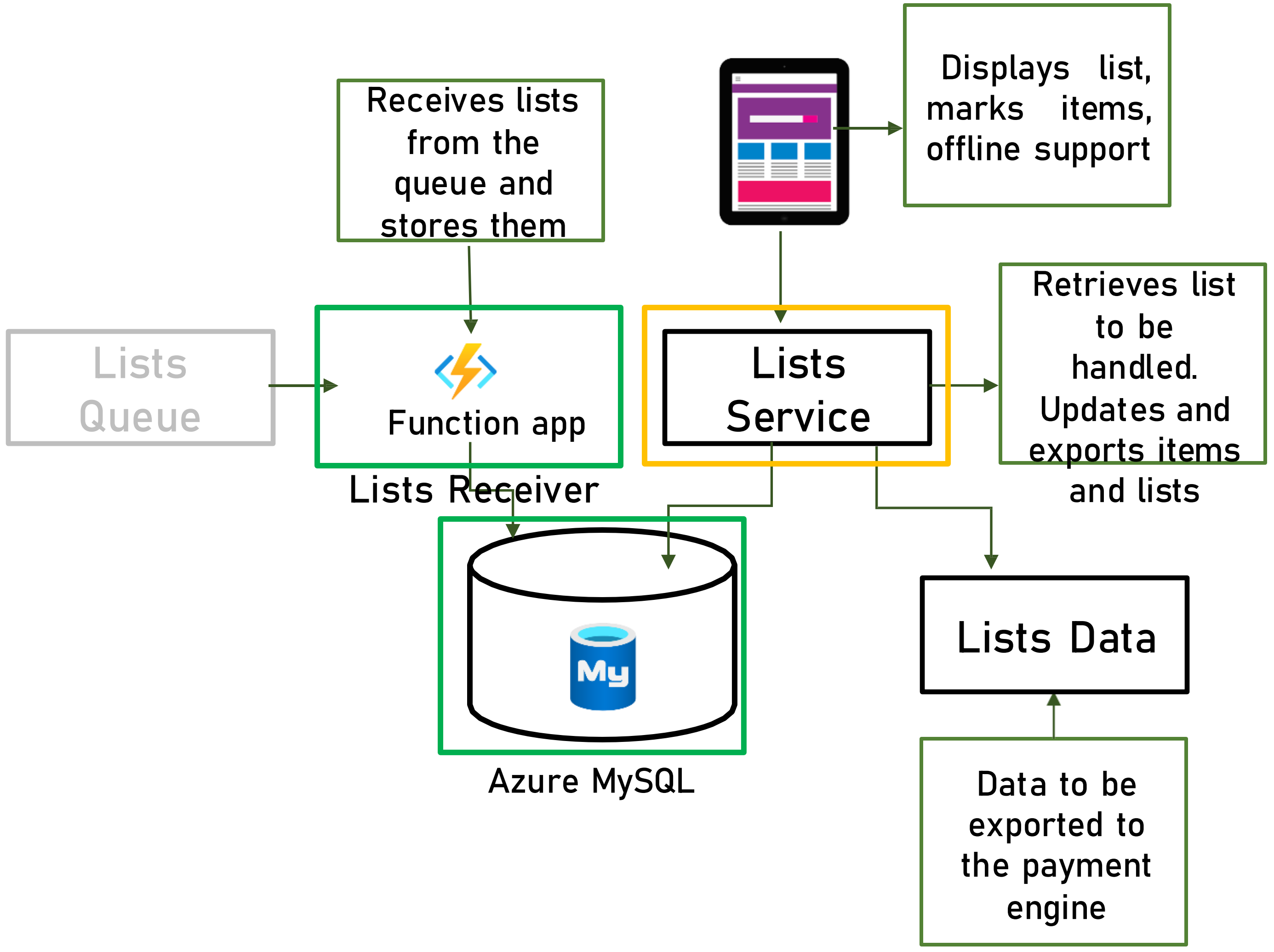
Components


Receives lists from the queue and stores them
Displays list, marks items, offline support
Retrieves list to be handled. Updates and exports items and lists
Lists Queue
Lists Receiver
Lists Service
Lists DB
Lists Data
Data to be exported to the payment engine

Components


Receives lists from the queue and stores them
Displays list, marks items, offline support
Lists Queue
Function app
Lists Receiver
Lists Service
Retrieves list to be handled. Updates and exports items and lists
Azure MySQL
Lists Data
Data to be exported to the payment engine

Components


Receives lists from the queue and stores them
Displays list, marks items, offline support
Lists Queue
Function app
Lists Receiver
Lists Service
Retrieves list to be handled. Updates and exports items and lists
Azure MySQL
Lists Data
Data to be exported to the payment engine

Lists Service

What it does:

- Allows employees to query lists

- Marks items in list

- Exports payment data

Application Type

• Web App & Web API	✓
• Mobile App	✗
• Console	✗
• Service	✗
• Desktop App	✗

Technology Stack

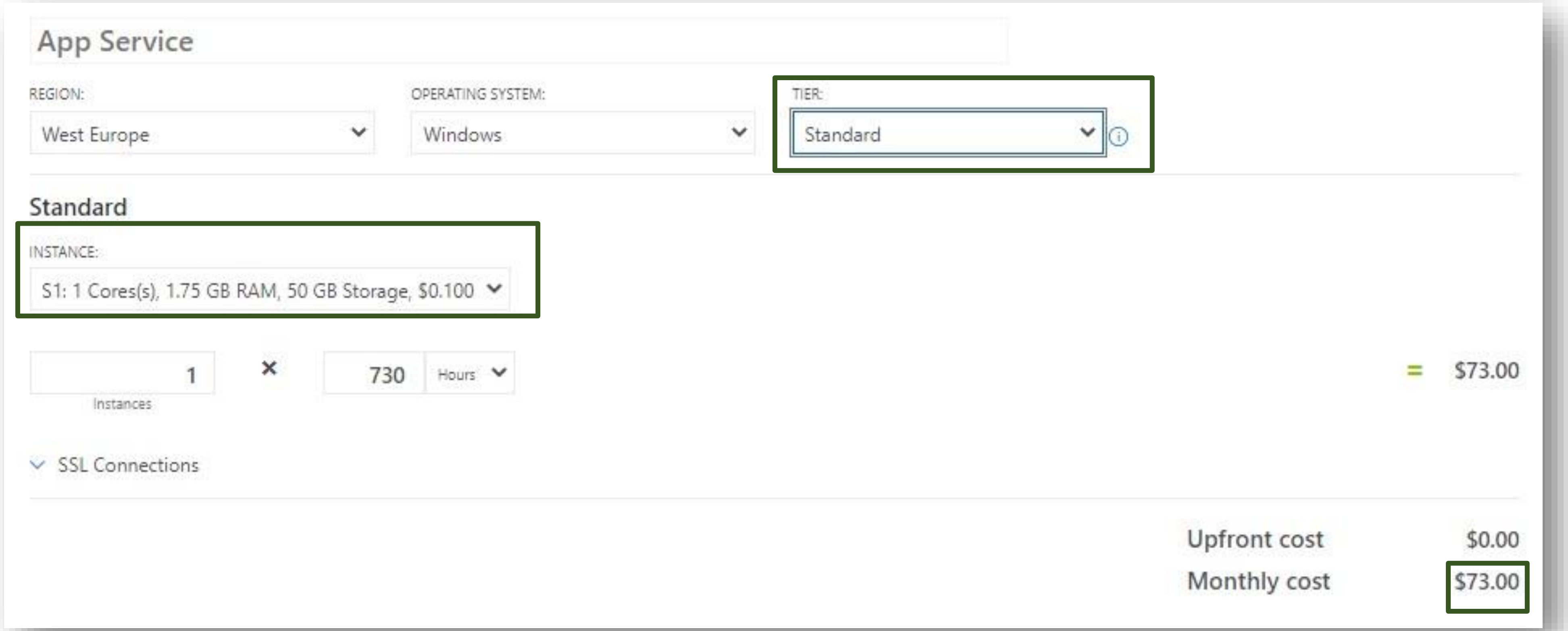
Java™

Azure Web App

App Service

- Fully managed web app & API
- Supports many platforms
- Autoscale
- Support for WebJobs

Azure Web App



App Service

REGION: West Europe

OPERATING SYSTEM: Windows

TIER: Standard

Standard

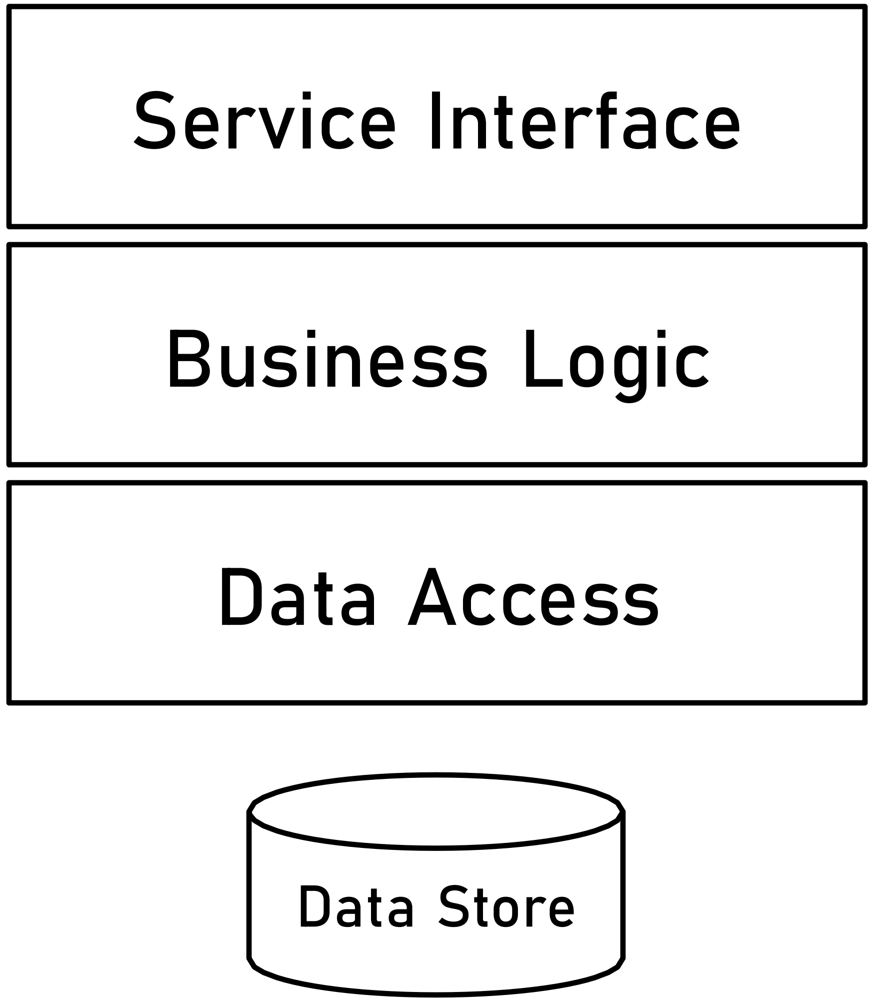
INSTANCE: S1: 1 Cores(s), 1.75 GB RAM, 50 GB Storage, $0.100

1 Instances × 730 Hours = $73.00

SSL Connections

Upfront cost	$0.00
Monthly cost	$73.00

Architecture



API

- Get next list to be processed (by location)
- Mark item as collected / unavailable
- Export list's payment data

API

Functionality	Path	Return Codes

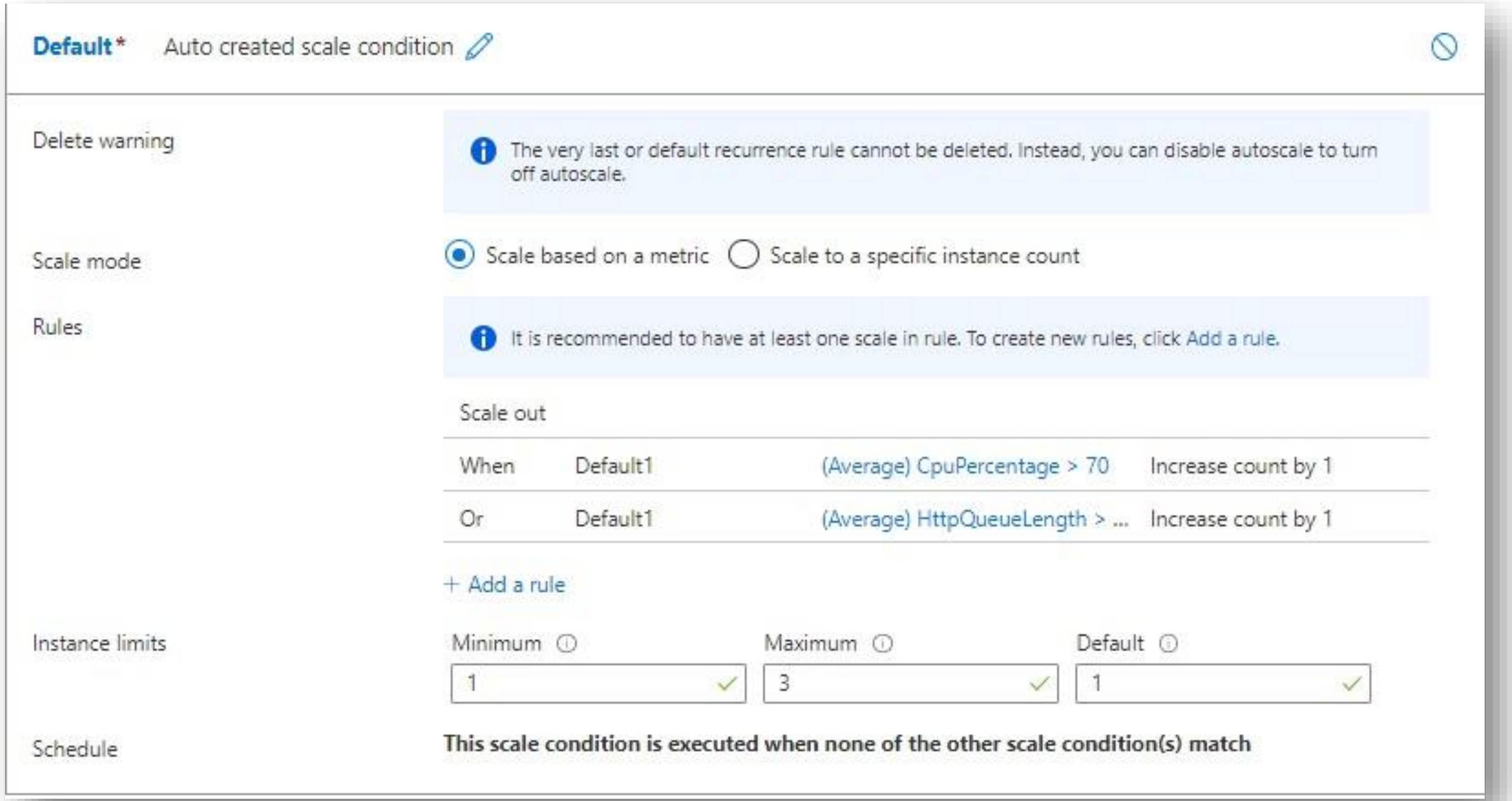
Get next list to be processed	`GET /api/v1/lists/next?location=…`	`200 OK`
		`400 Bad Request`

Mark item as collected / unavailable	`PUT /api/v1/list/`*`{listId}`*`/item/`*`{itemId}`*	`200 OK`
		`404 Not Found`

Export list’s payment data	`POST /api/v1/list/`*`{listId}`*`/export`	`200 Ok`
		`404 Not Found`

Lists Service Redundancy

App service auto



Default * Auto created scale condition

Delete warning: The very last or default recurrence rule cannot be deleted. Instead, you can disable autoscale to turn off autoscale.

Scale mode: [✓ Scale based on a metric] Scale to a specific instance count

Rules: It is recommended to have at least one scale in rule. To create new rules, click Add a rule.

Scale out

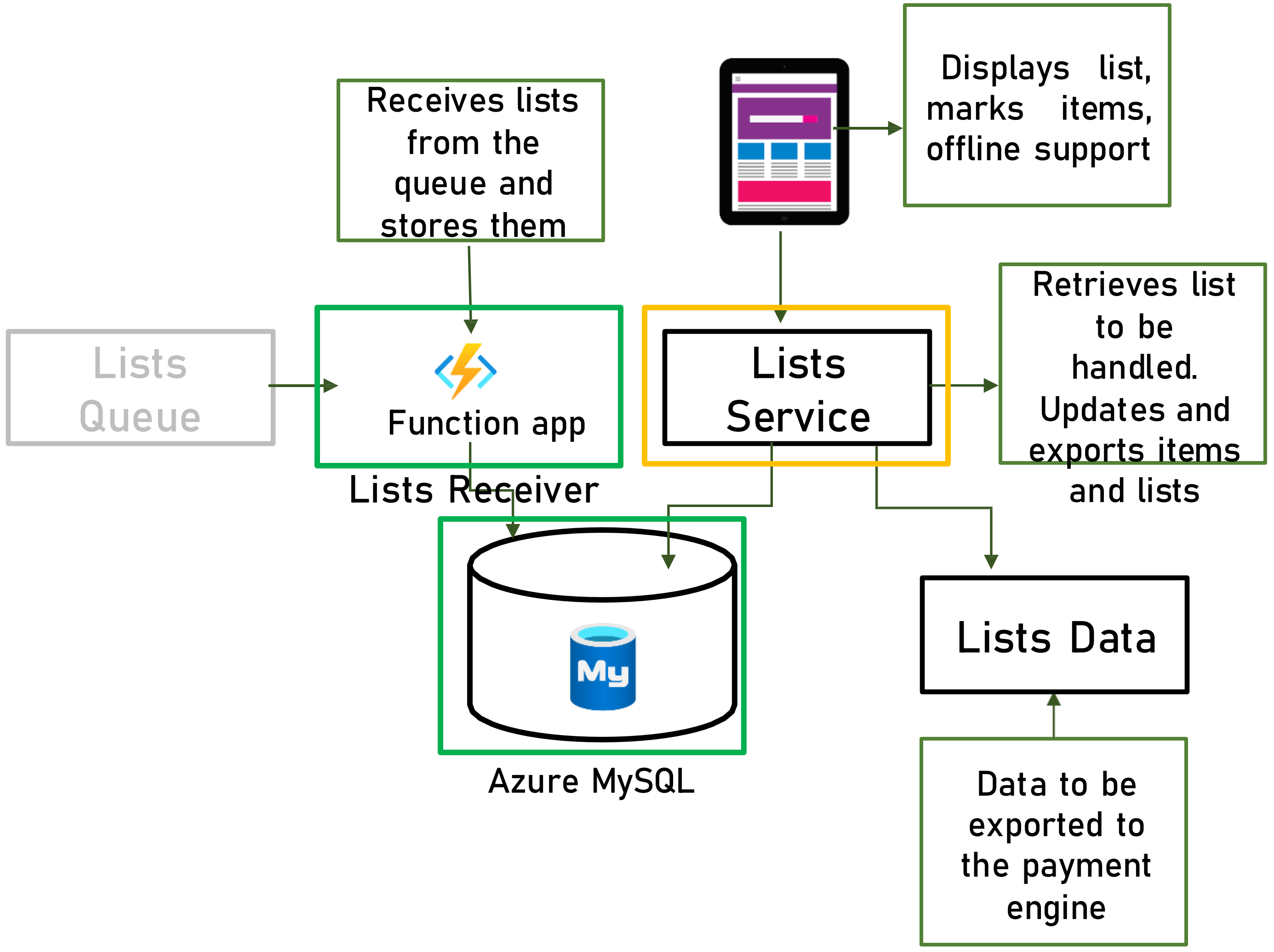
When	Default1	(Average) CpuPercentage > 70	Increase count by 1
Or	Default1	(Average) HttpQueueLength > ...	Increase count by 1

+ Add a rule

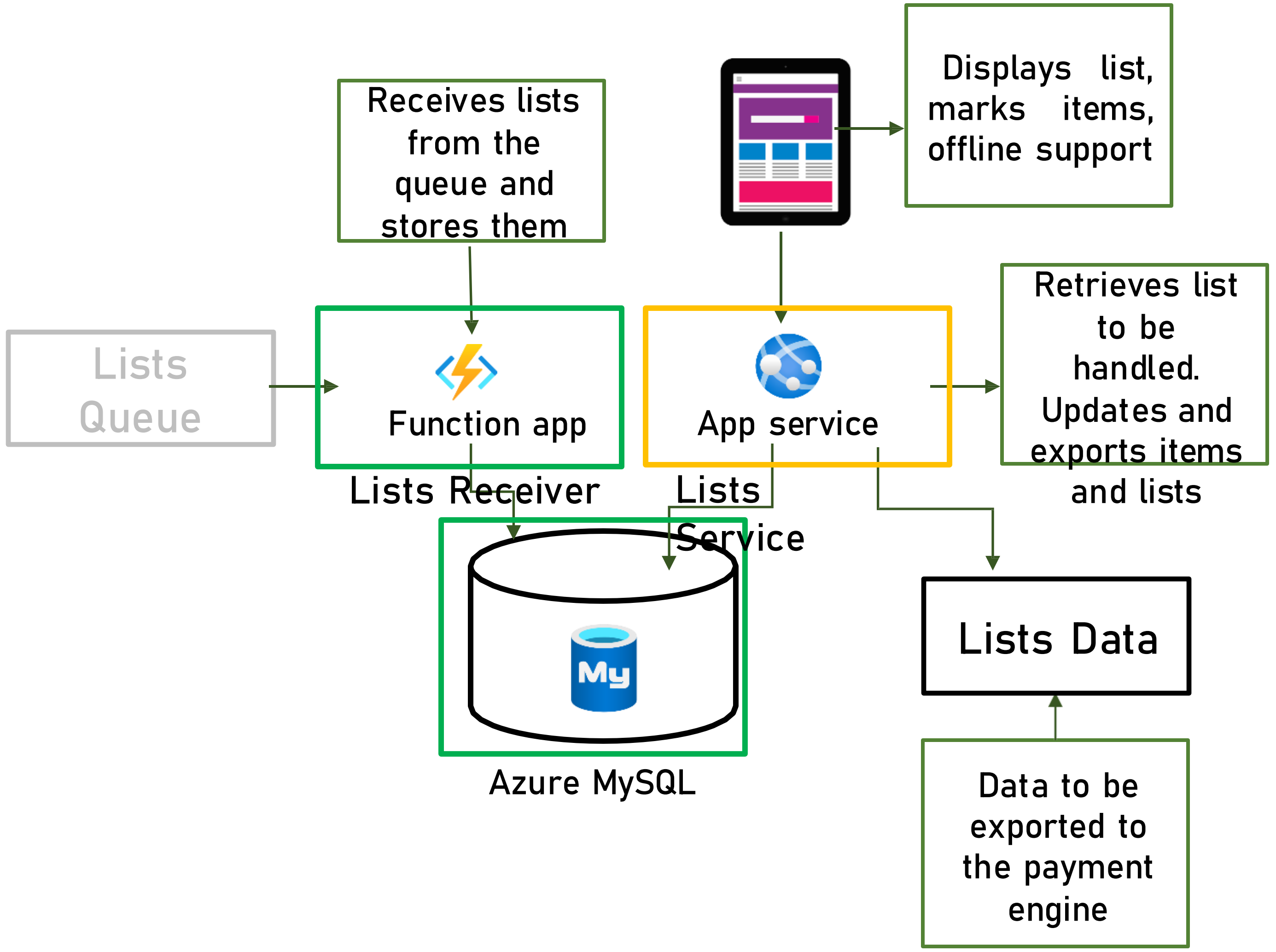
Instance limits: Minimum 1, Maximum 3, Default 1

Schedule: **This scale condition is executed when none of the other scale condition(s) match**

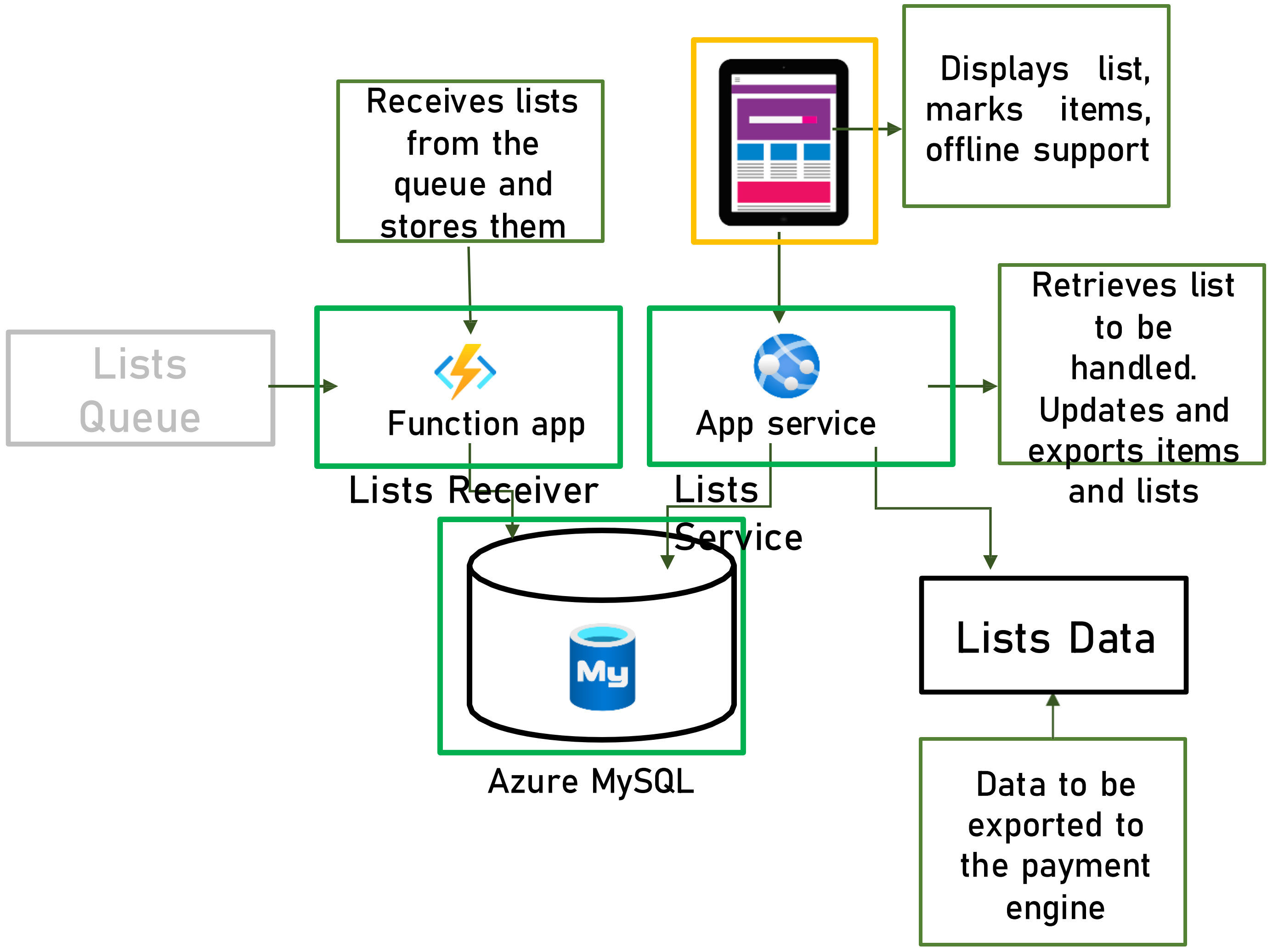
Components


Receives lists from the queue and stores them
Displays list, marks items, offline support
Retrieves list to be handled. Updates and exports items and lists
Lists Queue
Function app
Lists Receiver
Lists Service
Azure MySQL
Lists Data
Data to be exported to the payment engine

Components


Receives lists from the queue and stores them
Displays list, marks items, offline support
Lists Queue
Function app
Lists Receiver
App service
Lists Service
Retrieves list to be handled. Updates and exports items and lists
Azure MySQL
Lists Data
Data to be exported to the payment engine

Components


Lists Queue
Receives lists from the queue and stores them
Function app
Lists Receiver
Displays list, marks items, offline support
App service
Lists Service
Retrieves list to be handled. Updates and exports items and lists
Azure MySQL
Lists Data
Data to be exported to the payment engine

Front End

What it does:

- Displays shopping list

- Marks items as unavailable / collected

- Sends list to payment system

- Supports offline mode

Application Type

- Web App & Web API ✗
- Mobile App ✓
- Console ✗
- Service ✗
- Desktop App ✓

Technology Stack

Need to decide between:

Desktop, windows based (WPF) Native)

- Supports all OS functionalities
- Utilizes other apps on the machine (ie. DB)
- Requires setup, Windows

Web based (Electron, React

- Limited functionality
- Cannot use other apps
- Fully compatible with other form factors (phones, etc.)
- No setup required
- Cheaper hardware

Technology Stack

Need to decide between:

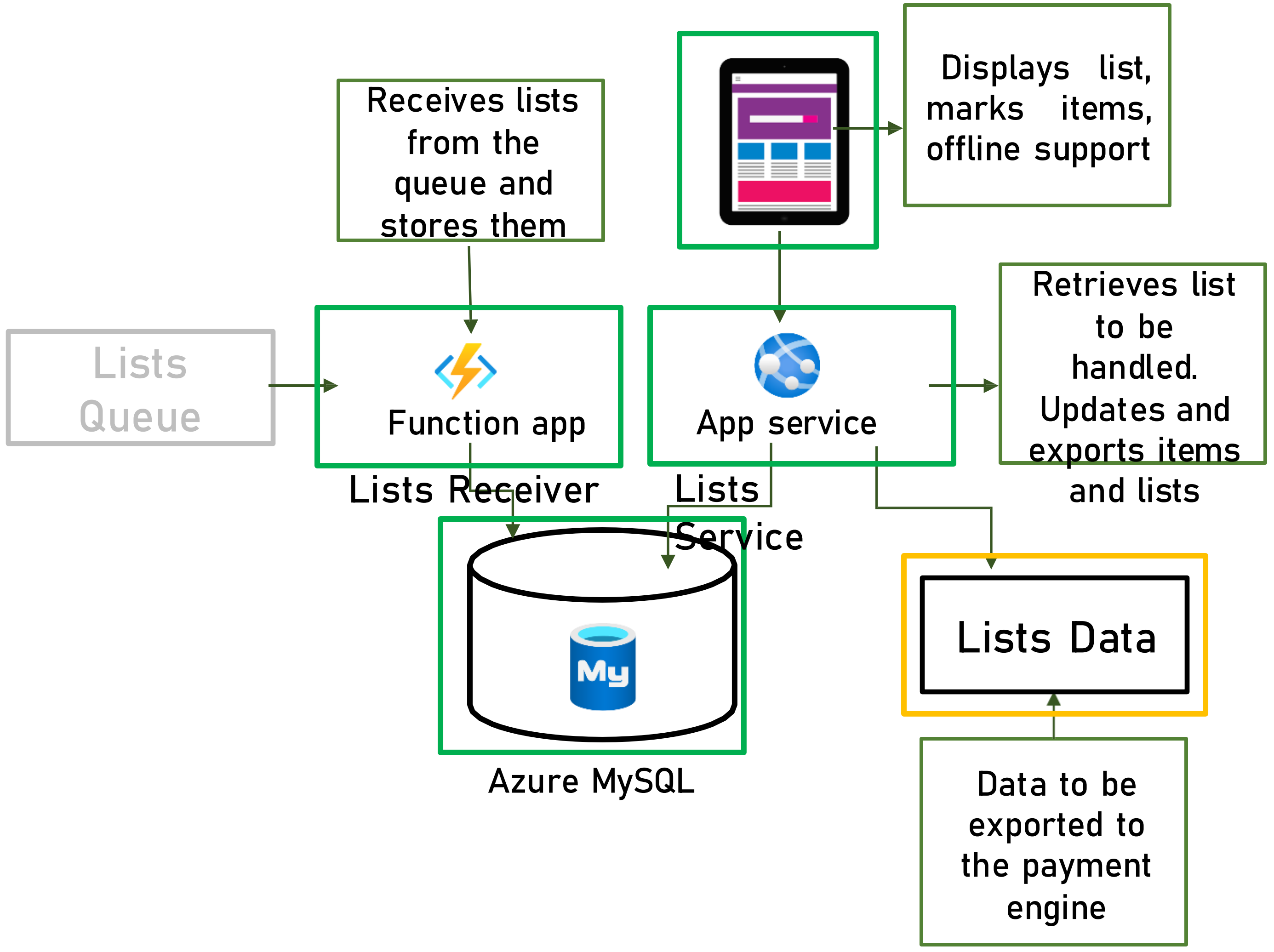
Web based (Electron, React Native)

- Limited functionality
- Cannot use other apps
- Fully compatible with other forms (phones, etc.)
- No setup required
- Cheaper hardware

Front End Redundancy

Not Relevant…

Components


Receives lists from the queue and stores them
Displays list, marks items, offline support
Lists Queue
Function app
Lists Receiver
App service
Lists Service
Retrieves list to be handled. Updates and exports items and lists
Azure MySQL
Lists Data
Data to be exported to the payment engine

Export Lists Data

What it does:

- Used to send shopping lists' data to payment

  system

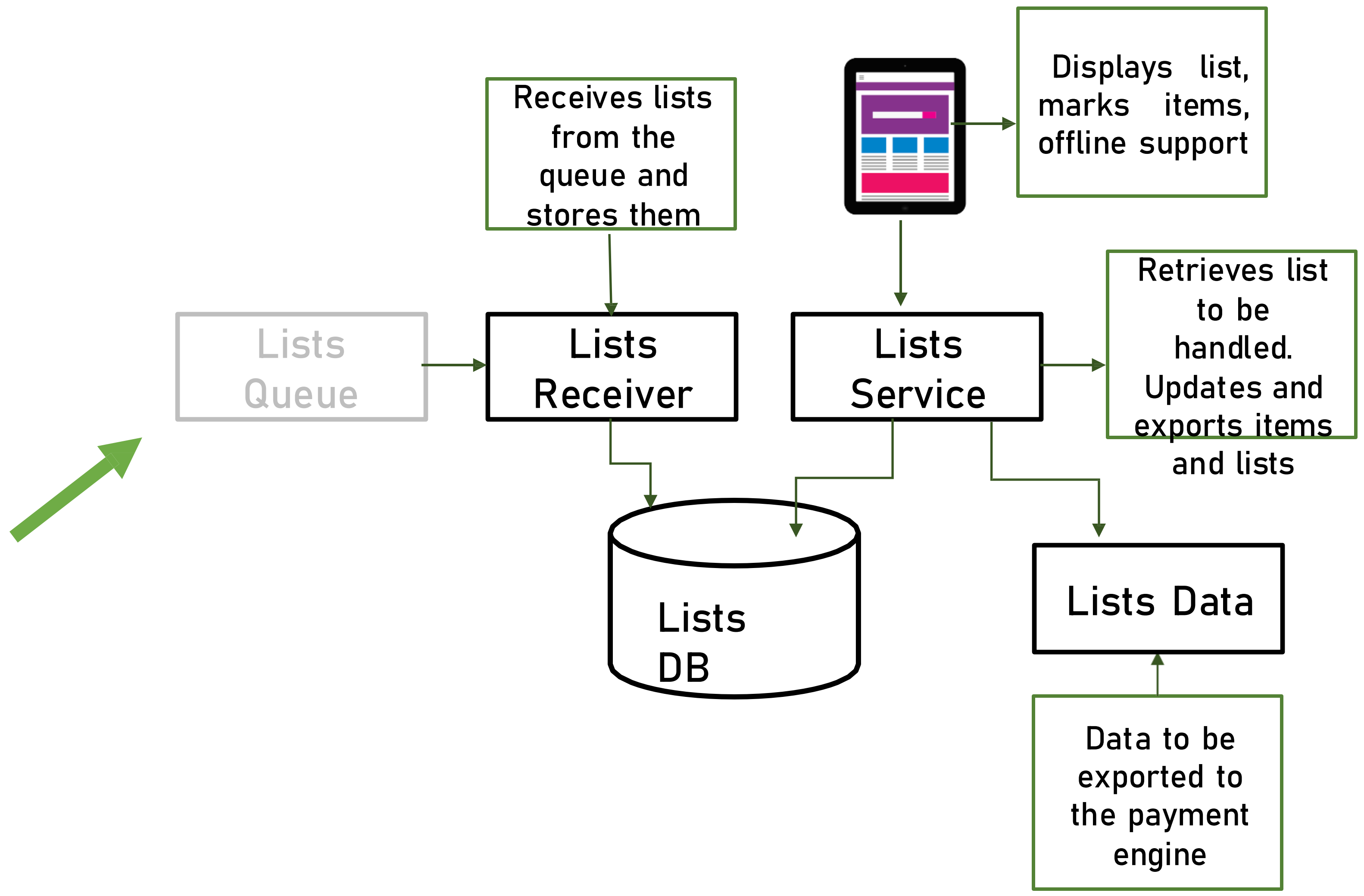
- Basically – a queue

Export Lists Data- Questions

1. Is there an existing queue mechanism in the company?
2. Develop our own or use 3rd party?

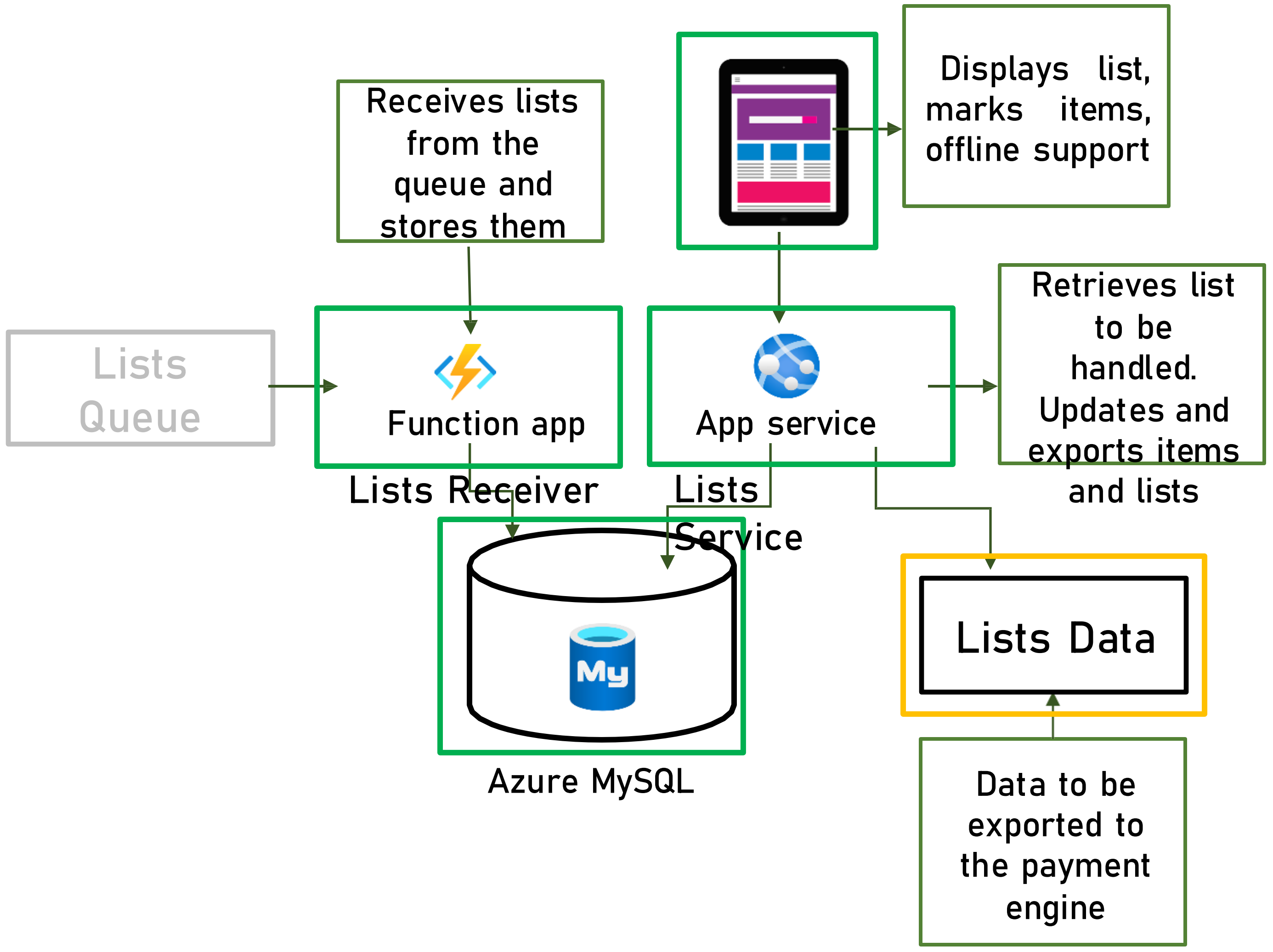
Yes

Components

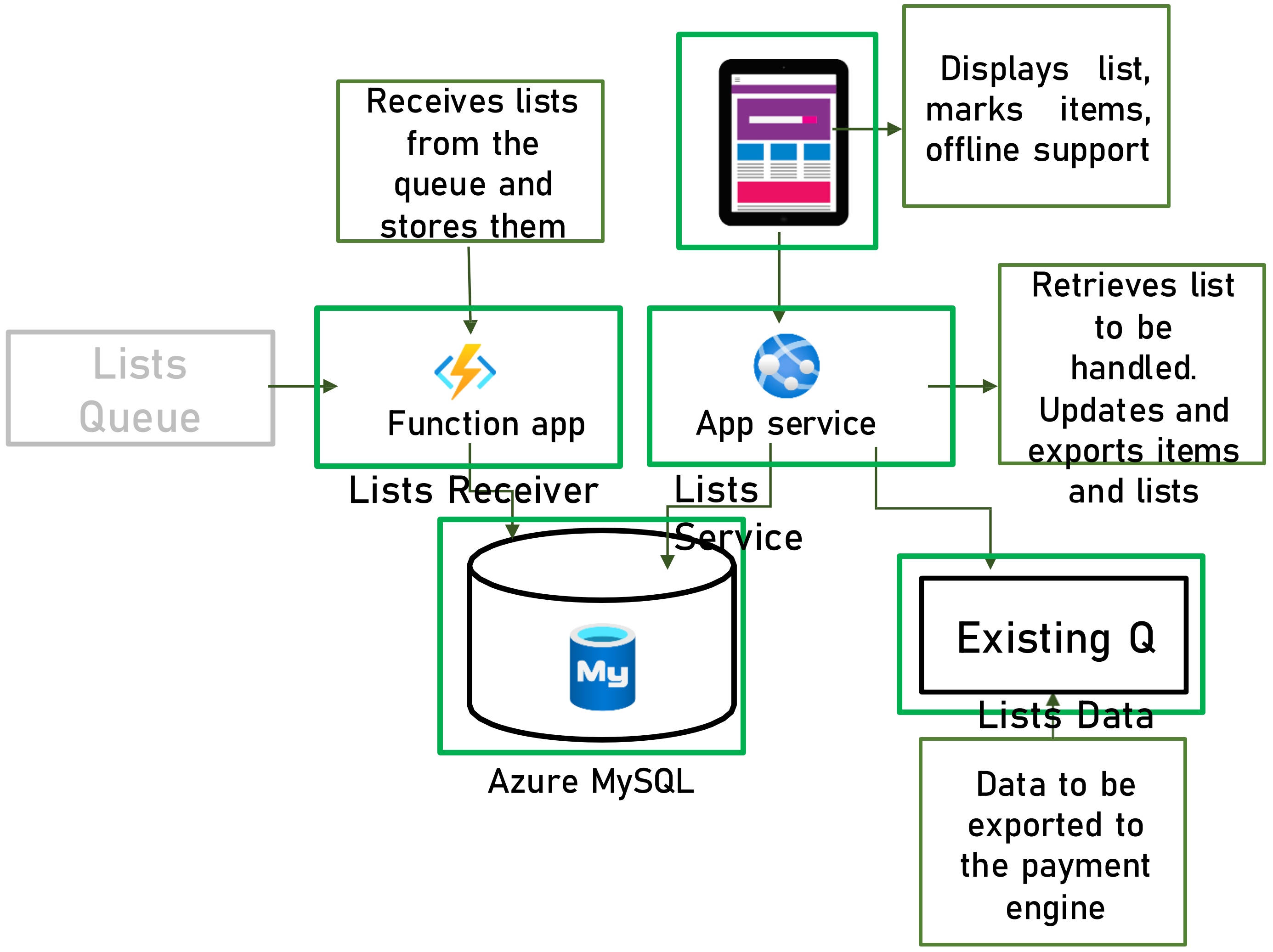


Lists Queue
Lists Receiver
Receives lists from the queue and stores them
Lists Service
Displays list, marks items, offline support
Retrieves list to be handled. Updates and exports items and lists
Lists DB
Lists Data
Data to be exported to the payment engine

Components


Receives lists from the queue and stores them
Displays list, marks items, offline support
Lists Queue
Function app
Lists Receiver
App service
Lists Service
Retrieves list to be handled. Updates and exports items and lists
Azure MySQL
Lists Data
Data to be exported to the payment engine

Components


Receives lists from the queue and stores them
Displays list, marks items, offline support
Lists Queue
Function app
Lists Receiver
App service
Lists Service
Retrieves list to be handled. Updates and exports items and lists
Azure MySQL
Existing Q
Lists Data
Data to be exported to the payment engine

Security

- Pay attention to:
  - Public accessible databases
  - Unprotected access to App Service

Security

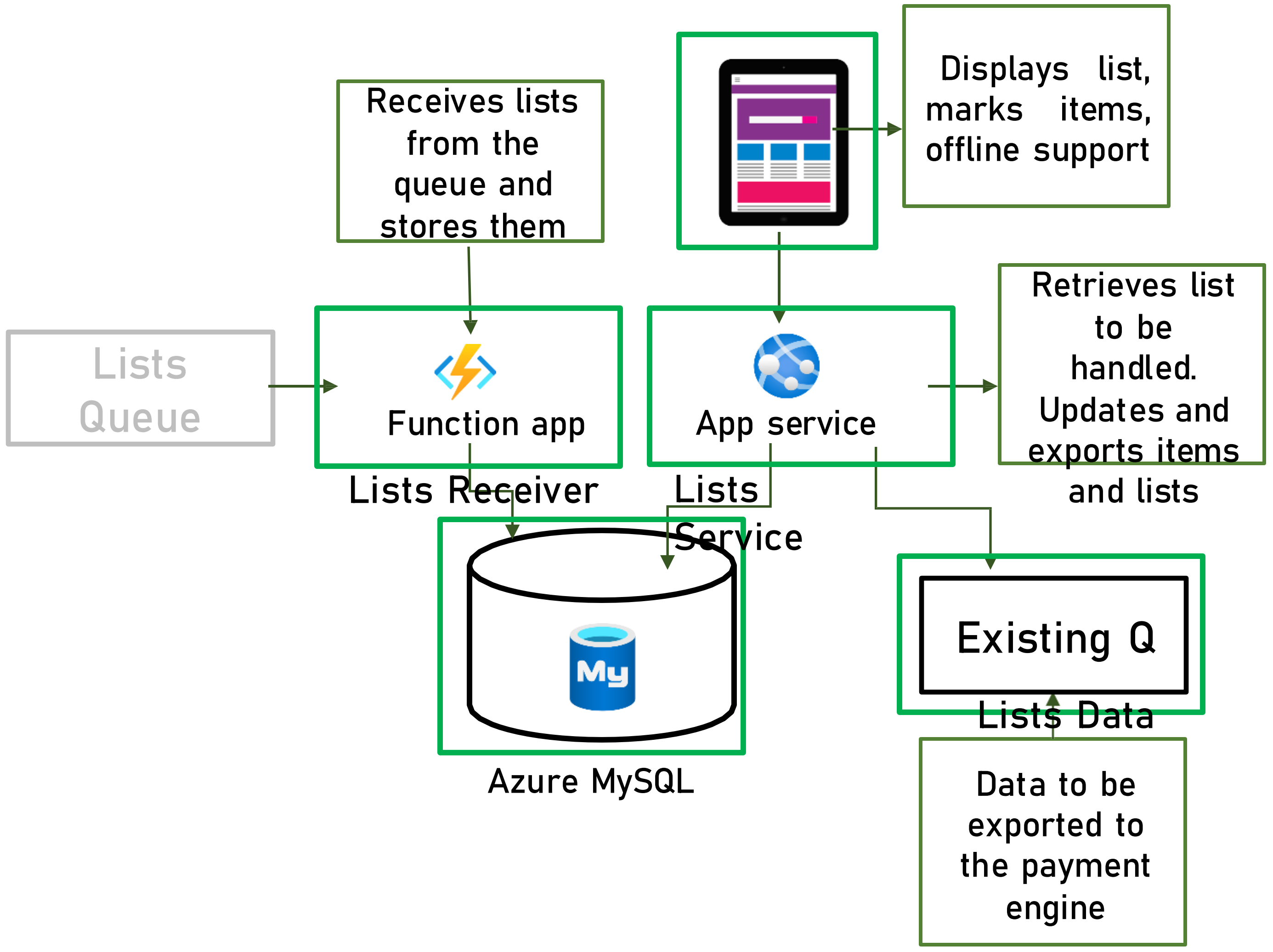
- To-Do:
  - Block access to databases from unauthorized

    IP addresses

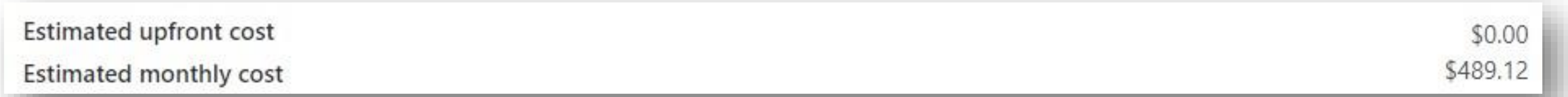
Security

- What about the App Service?
  - The client decided not to place WAF in front the App Service
    - Small service
    - Read-only operations
    - Save costs

Architecture Diagram


Receives lists from the queue and stores them
Displays list, marks items, offline support
Retrieves list to be handled. Updates and exports items and lists
Lists Queue
Function app
App service
Lists Receiver
Lists Service
Azure MySQL
Existing Q
Lists Data
Data to be exported to the payment engine

Cost



Estimated upfront cost	$0.00
Estimated monthly cost	$489.12

Download detailed cost estimation from the lecture's resources