

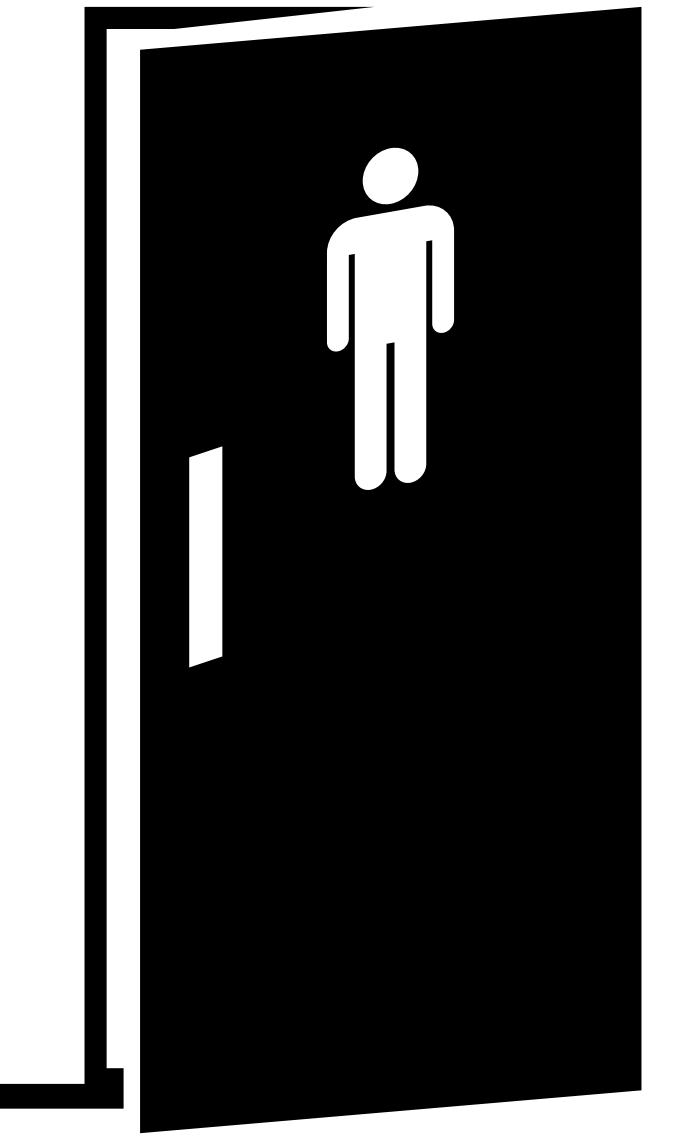


peepal

SC2006 FDAB Team 1

Soong Jun Shen
Adam Soh Shi Jie
Joyce Lee Jia Xuan

Liew Jia Wei
Tan Kai Hooi





peepal

Problem Statement

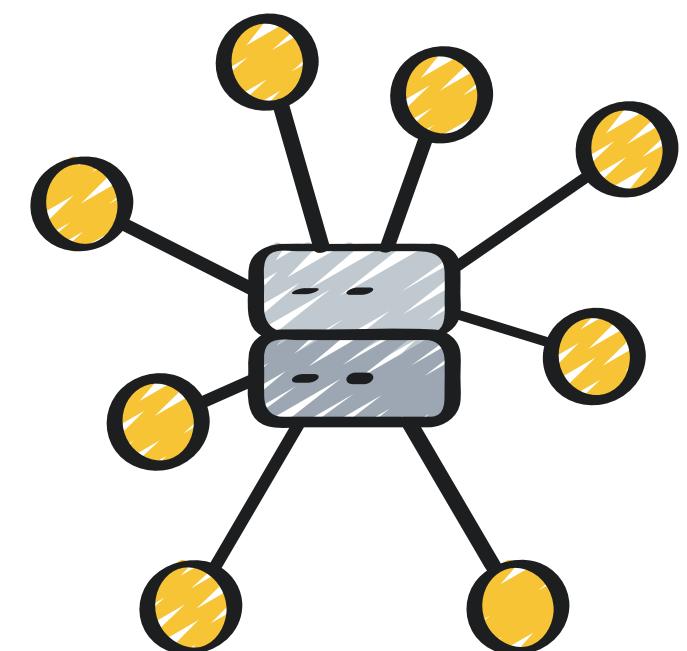


Finding a **clean and accessible public toilet** is often a challenge in busy cities like Singapore. People struggle to identify nearby toilets with essential features such as **bidets, accessibility for people with disabilities, showers and sanitisers.**



peepal

What if we can use a **centralised, easily accessible application** to address this issue?



Introduce our solution



peepal

A Community-Driven Crowd Sourced Toilet Finder



peepal

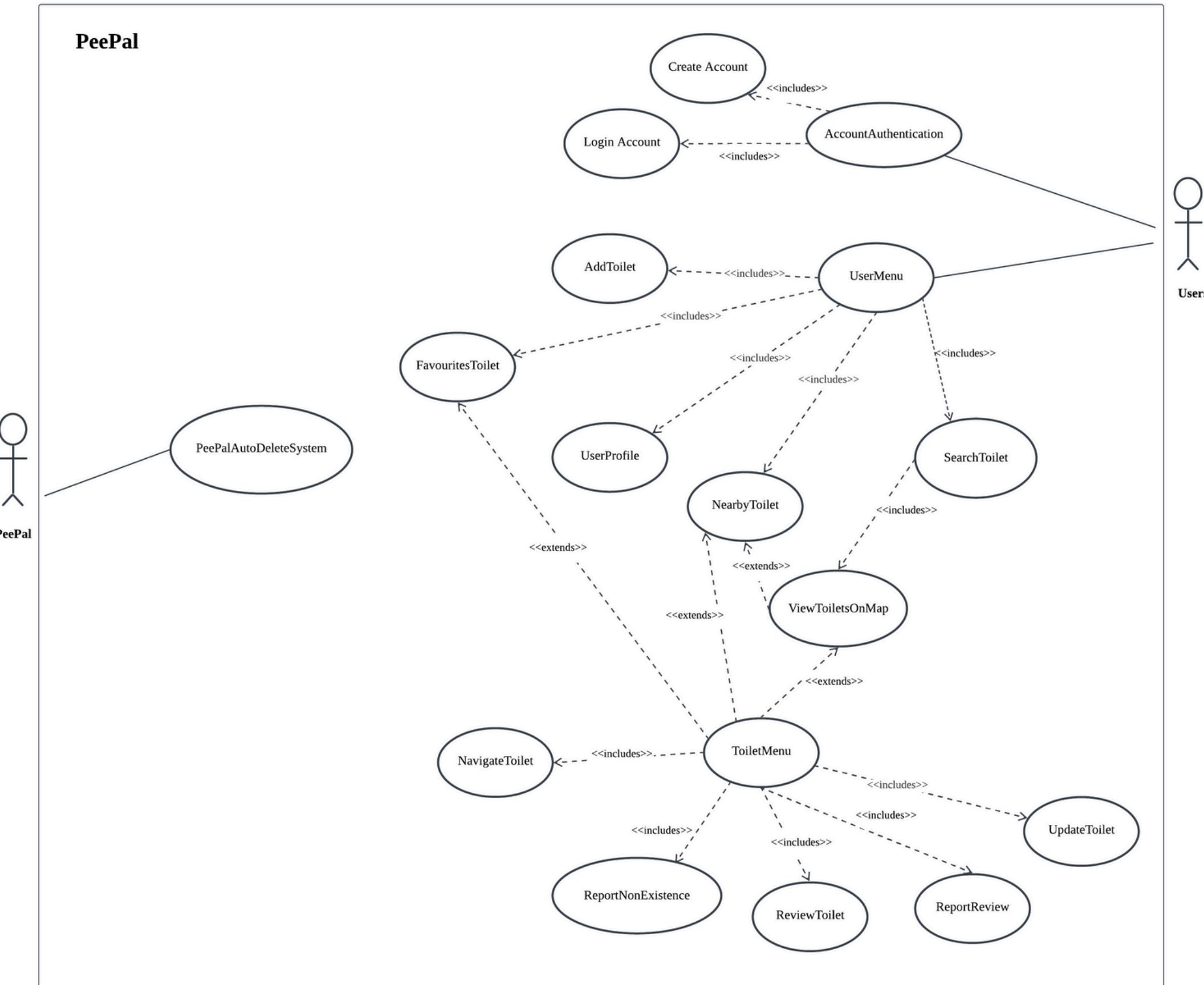
Key features:

- Find the **nearest public toilets** based on your current location and receive turn-by-turn directions to **navigate** to any toilet..
- Allow users to identify toilets with specific features, such as **bidets, showers, sanitiser, and OKU-friendly facilities** across Singapore.
- View **ratings and reviews** from other **PeePal** users to make informed decisions about toilet quality and cleanliness.
- **Community-driven data** to ensure data remains open, transparent and free for all users.

Use case diagram



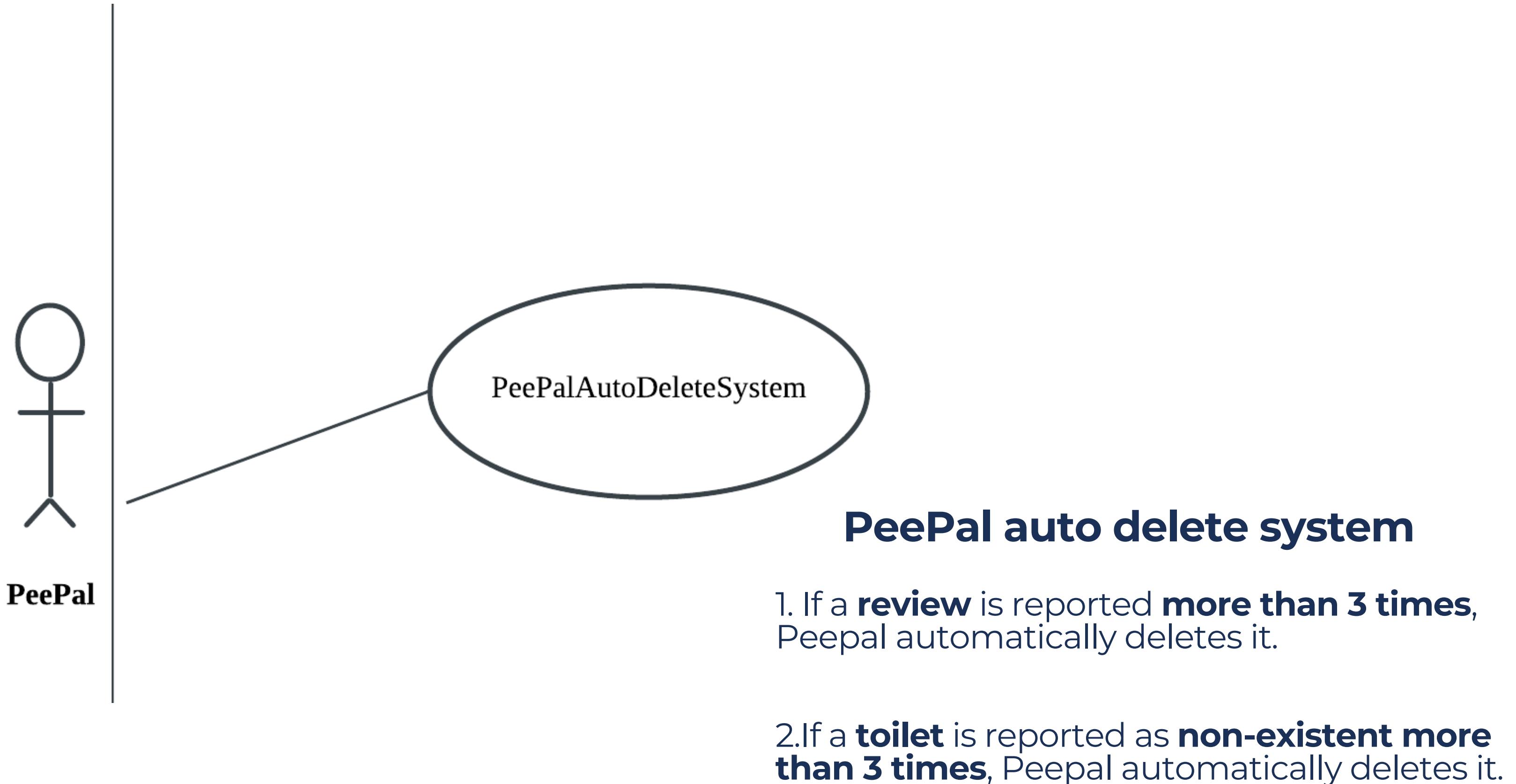
peepal



Use case diagram



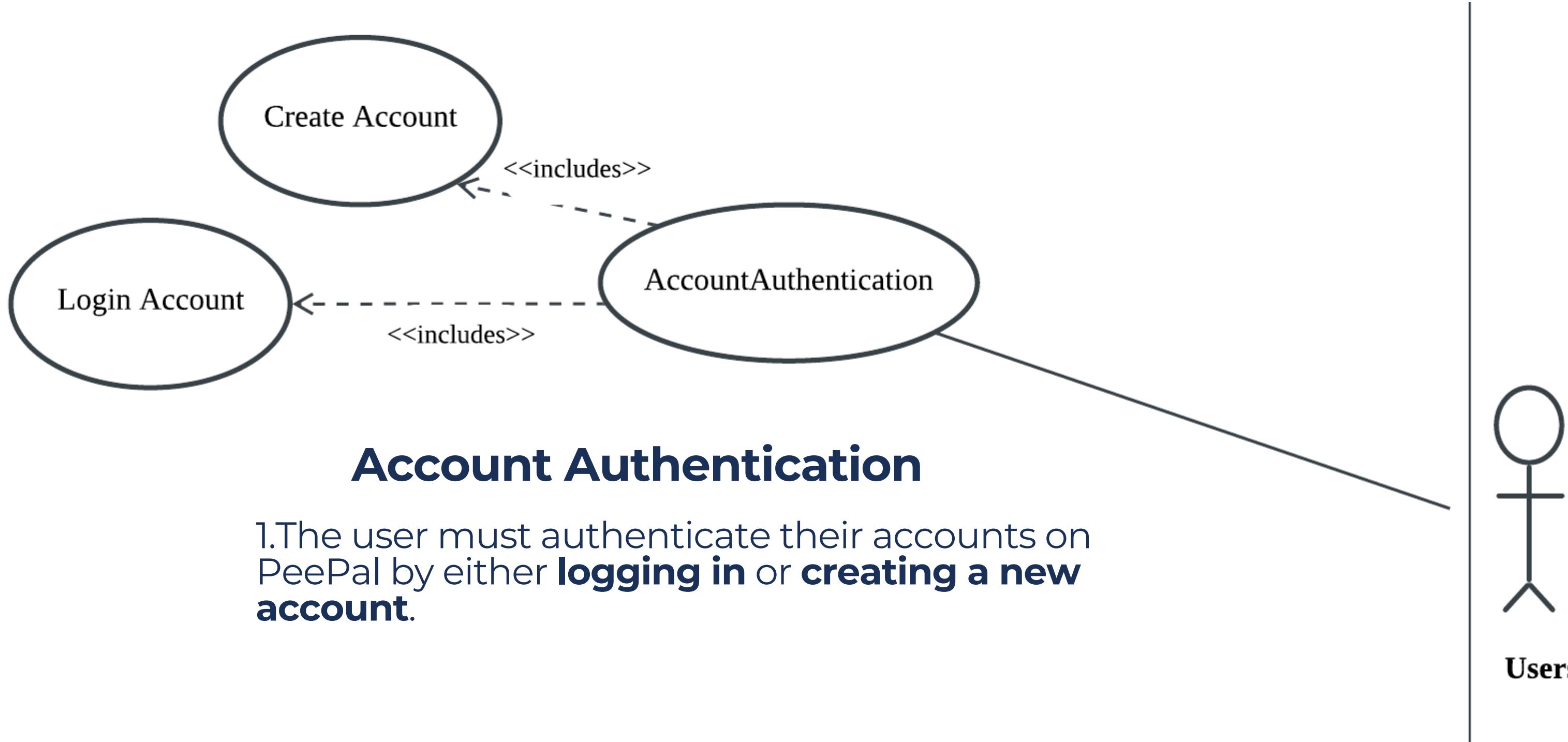
peepal



Use case diagram



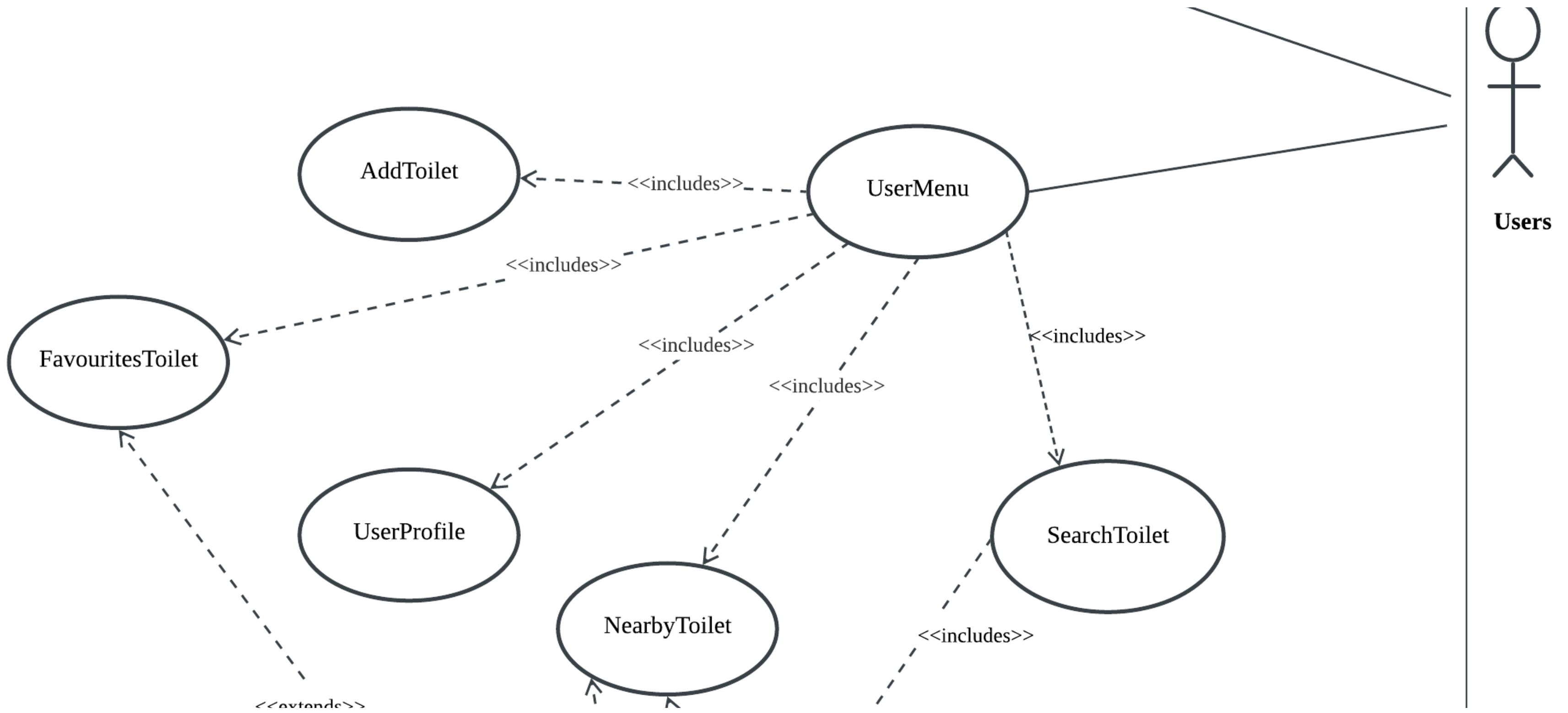
peepal



Use case diagram



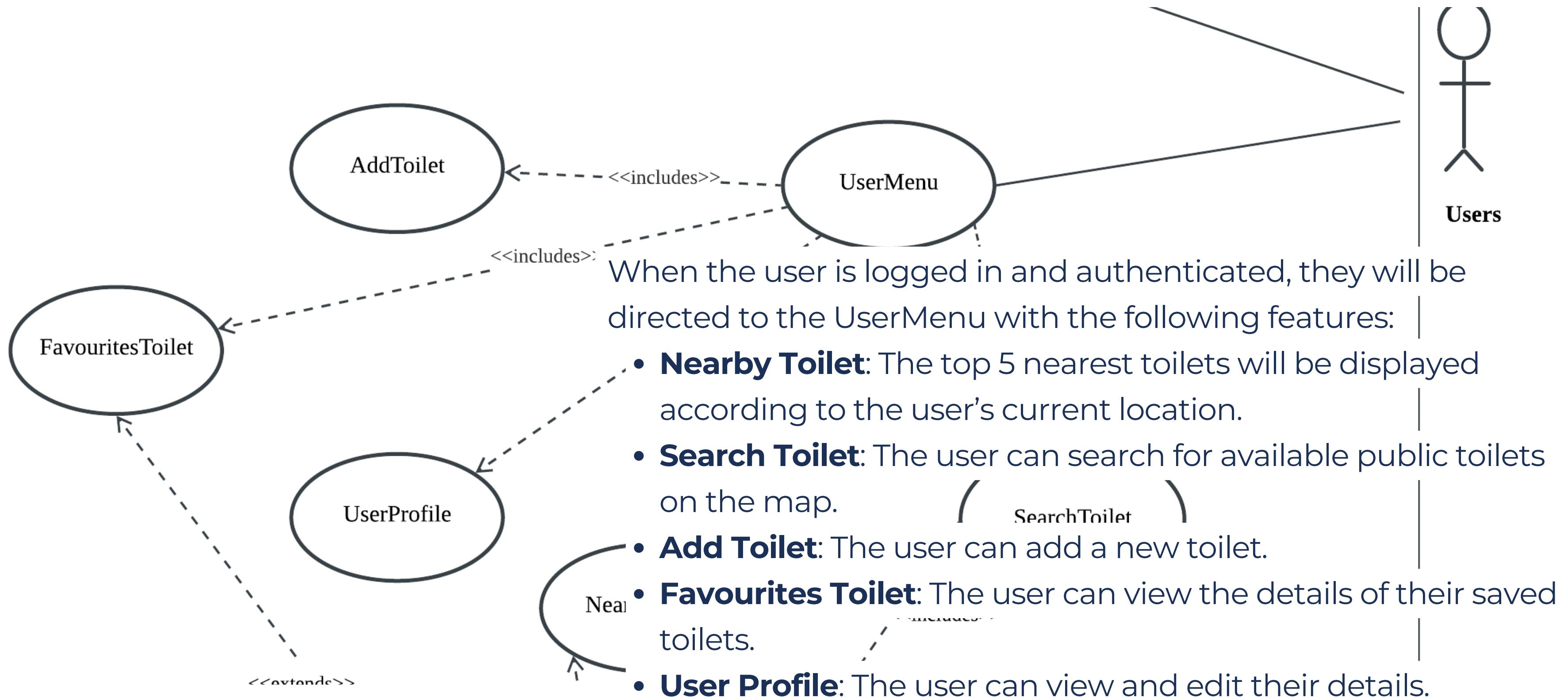
peepal



Use case diagram



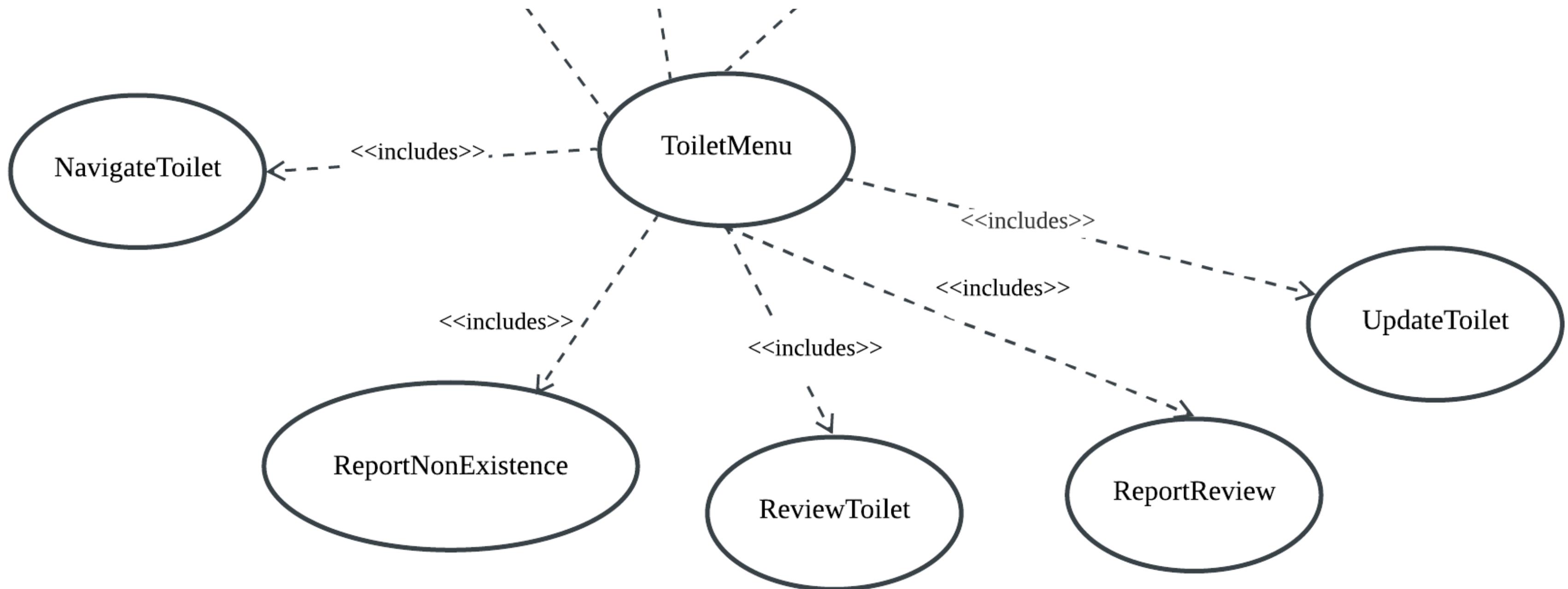
peepal



Use case diagram



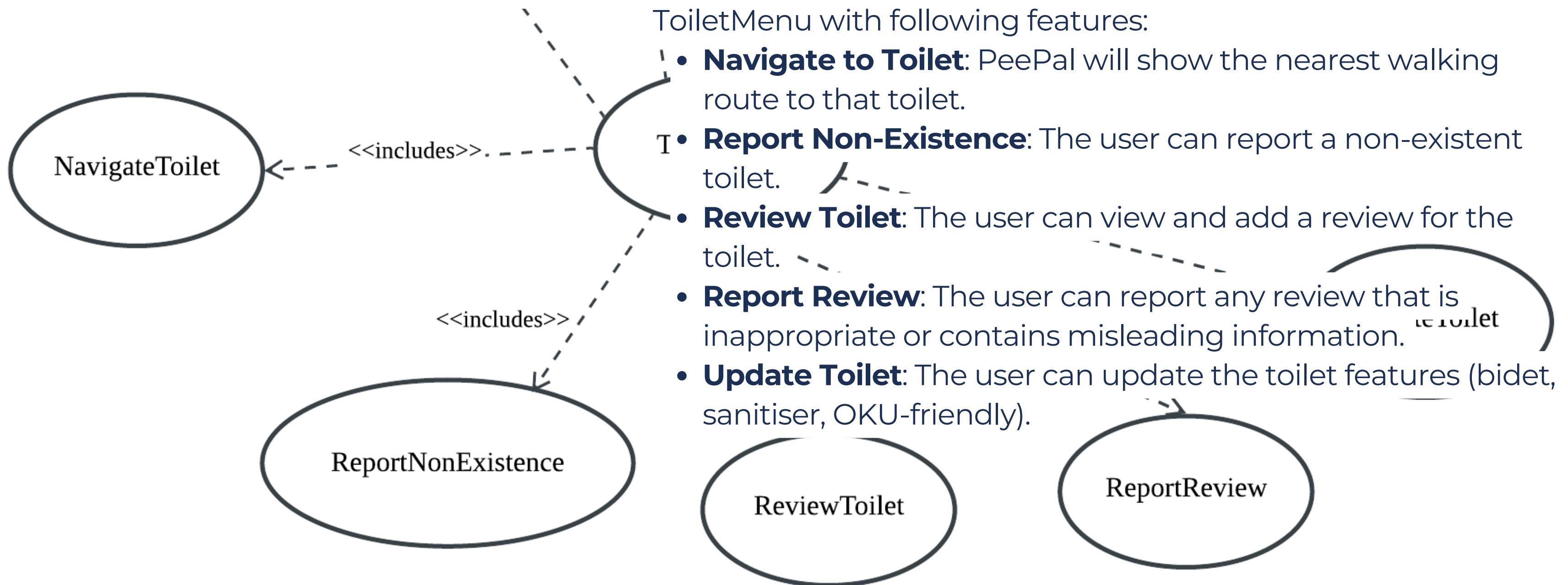
peepal



Use case diagram



peepal



External APIs

Google Maps API

Used the Nearby Places API, Places Photo API and Places Details API to retrieve and index the data on public toilets across Singapore.

Apple MapKit API

Provides real-time navigation and ETA matrix computations to the nearest toilets, and interactive map interface.

S3 Storage Buckets

For permanent object storage using the standard S3 Bucket APIs.



peepal

Live Demo!

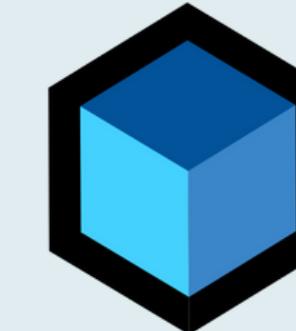


peepal

Tech Stack



Flutter



bloc

state management, layered architecture using repository pattern



Drizzle ORM



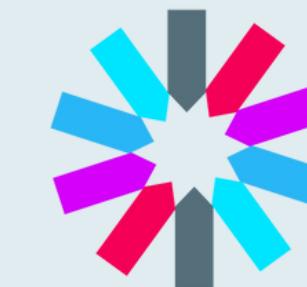
Hono



PostgreSQL

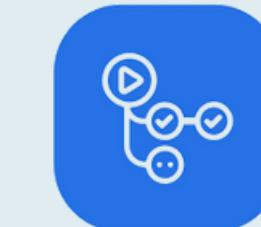


VITEST



JWT Based Auth

Custom Backend Stack with Test-Driven Development



GitHub Actions

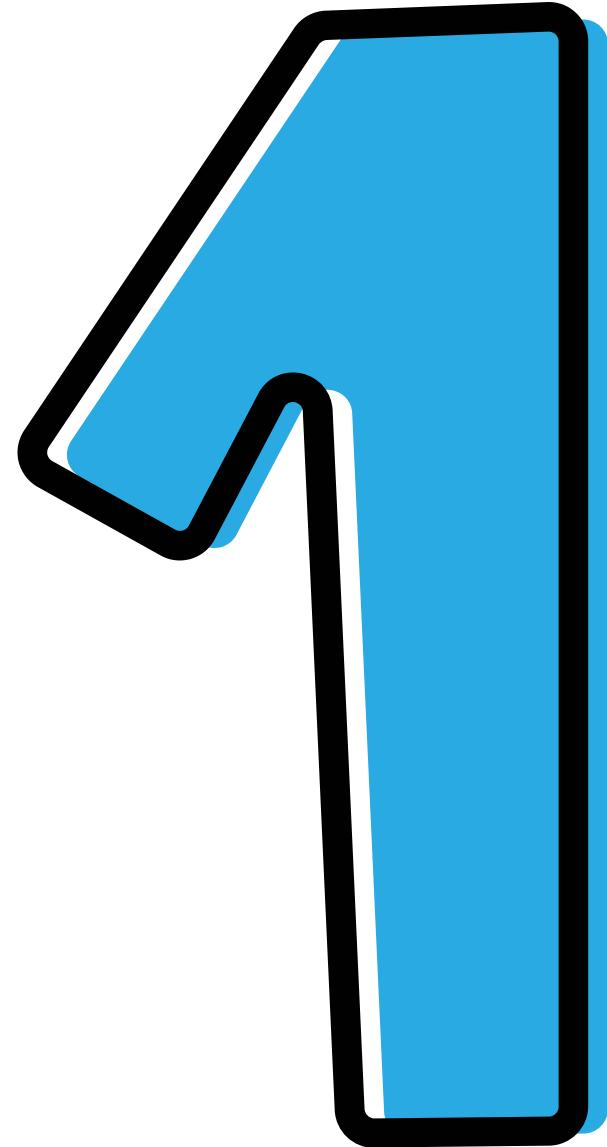


Sevalla

Continuous Integration and Deployment



peepal



Good SWE Practices



peepal

Good SWE Practices



1. Documentation
2. Consistent Code Style & Naming Convention
3. Reusability and Refactoring
4. Waterfall structure



peepal

1. Documentation - README

2006-FDAB-P1

Peepal - Toilet Finder Application

Overview

Peepal is a modern toilet finder application that helps users locate nearby toilets with detailed information and reviews. The application is built using a microservices architecture with separate frontend and backend services.

Tech Stack

Backend

- **Framework:** Hono (TypeScript)
- **Database:** PostgreSQL with Drizzle ORM
- **Authentication:** JWT
- **Image Storage:** MinIO
- **Testing:** Vitest
- **Validation:** Zod
- **API Documentation:** Swagger/OpenAPI

Frontend

- **Framework:** Flutter
- **State Management:** BLoC Pattern
- **Navigation:** Flutter Navigation
- **UI Components:** Custom widgets with Material Design
- **Maps:** MapKit API
- **Location Services:** Core Location Framework

Architecture & Design Patterns

Backend Architecture

1. RESTful API Design
 - Clean and consistent API endpoints
 - HTTP status codes for error handling
 - JSON responses with proper validation
2. Database Design
 - Relational database with proper indexing
 - Foreign key constraints for data integrity
 - Separate tables for users, toilets, reviews, and favorites
3. Authentication Flow
 - JWT-based authentication
 - Secure password hashing with bcrypt
 - Role-based access control

Frontend Architecture

1. BLoC Pattern
 - Separation of concerns between UI and business logic
 - State management through BLoC classes
 - Event-driven architecture for UI updates
2. Widget Architecture
 - Reusable widgets for common UI components
 - Custom widgets for specific features
 - Responsive design for different screen sizes



1. Documentation - Code Comments

```
108
109  /**
110   * Creates a new toilet based on the provided information.
111   *
112   * The request body is validated using the defined schema to ensure correct data types are used.
113   *
114   * The location field is converted to a spatial point and inserted into the database.
115   *
116   * Logs the creation operation and returns the created toilet information upon success.
117   *
118   * @param c - The context containing request and response objects, and logger.
119   * @returns A JSON response with the created toilet details or an error message.
120   */
121  toiletApi.post('/create', validator('json', createToiletSchema), async (c) => {
122    const logger = c.get('logger')
123    const { name, address, latitude, longitude, location, handicapAvail, bidetAvail, showerAvail } = c.req.valid('json')
124
125    // Generate a unique ID for the new toilet
126    const toiletId: string = nanoid()
127
128    const [ newToilet ] = await db.insert(toilets).values([
129      {
130        id: toiletId,
131        name,
132        address,
133        // Convert location to a spatial point IMPORTANT! Use SRID 4326
134        location: sql`ST_SetSRID(ST_MakePoint(${location.x}, ${location.y}), 4326)`,
135        handicapAvail,
136        bidetAvail,
137        showerAvail,
138        sanitiserAvail,
139        rating: sql`${rating}::decimal`
140      }
141    ]).returning()
```

```
backend > src > middleware > ts auth.ts > ...
1  import { Context, Next } from 'hono'
2  import jwt from 'jsonwebtoken'
3  import { JWTPayload } from '../types/auth'
4
5  const JWT_SECRET = process.env.JWT_SECRET || 'your-secret-key'
6  // %I for Command, %L for Cascade
7  /**
8   * Authentication middleware that checks if a valid JWT token is present in the Authorization header.
9   *
10  * If the token is present and valid, it decodes the token and sets the 'user' property on the context object.
11  * If the token is invalid or not present, it returns a 401 response.
12  *
13  * @param {Context} c - The Hono Context object.
14  * @param {Next} next - The next middleware function to call.
15  */
16  export async function authMiddleware(c: Context, next: Next) {
17    // Auth header format: Bearer <token>
18    const token = c.req.header('Authorization')?.split(' ')[1]
19
20    if (!token) {
21      return c.json({ error: 'No token provided' }, 401)
22    }
23
24    try {
25      const decoded = jwt.verify(token, JWT_SECRET) as JWTPayload
26      // Forward the request to the next handler with the decoded user
27      c.set('user', decoded)
28      await next()
29    } catch (error) {
30      // User is not authenticated, return 401
31      return c.json({ error: 'Invalid token' }, 401)
32    }
33  }
34
```



peepal

2. Consistent Code Style & Naming Convention

Front-end



Dart



Prettier

Back-end

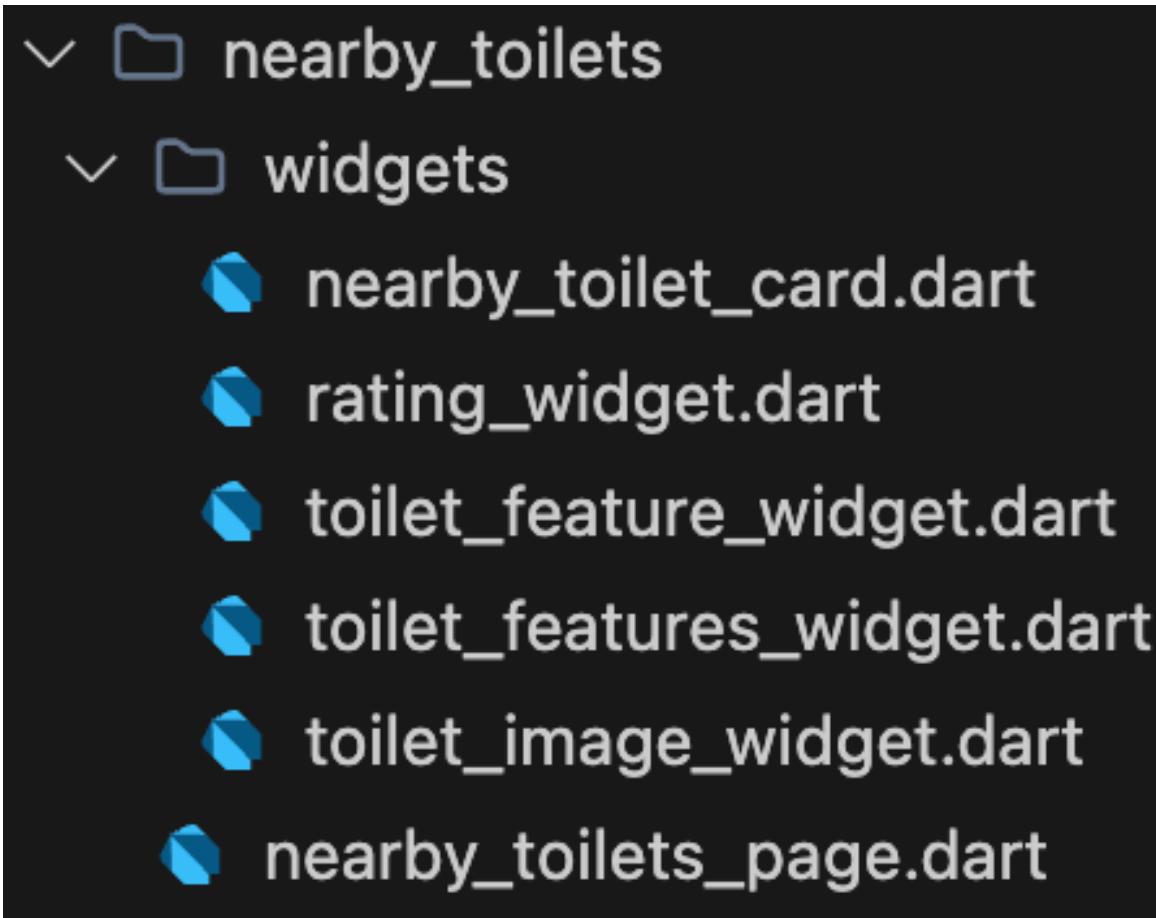


ESLint

Ensures that Code is readable, and maintainable, and understandable



3. Reusability & Refactoring

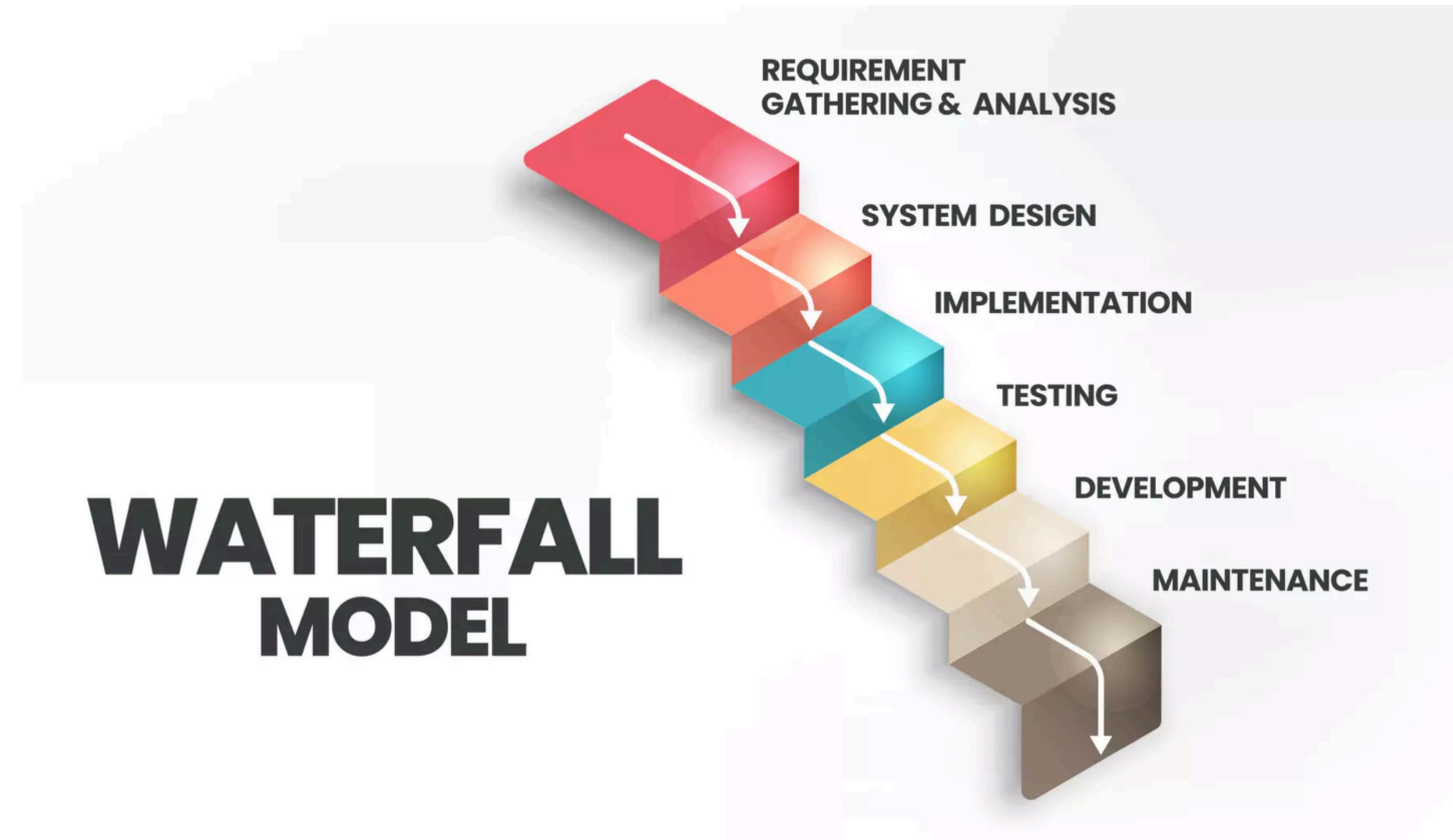


```
Row(
  mainAxisAlignment: MainAxisAlignment.spaceBetween,
  children: [
    ConstrainedBox(
      constraints: BoxConstraints(maxWidth: 240.0),
      child: AutoSizeText(
        toilet.name,
        style: const TextStyle(
          fontSize: 22.0, fontWeight: FontWeight.bold),
        maxLines: 2,
        overflow: TextOverflow.ellipsis,
      ),
    ),
    RatingWidget(rating: toilet.rating),
  ],
  SizedBox(height: 5.0),
  Text(
    toilet.address,
    maxLines: 2,
    style: TextStyle(
      height: 1.6,
      color: const Color(0xFF2D2D2D),
      fontSize: 14.0),
  ),
  SizedBox(height: 12.0),
  ToiletFeatureWidget(
    hasBidet: toilet.bidetAvail,
    hasOku: toilet.handicapAvail,
    hasShower: toilet.showerAvail,
    hasSanitizer: toilet.sanitiserAvail,
  ),
  const Spacer(),
)
```



peepal

4. Waterfall Structure





peepal

2

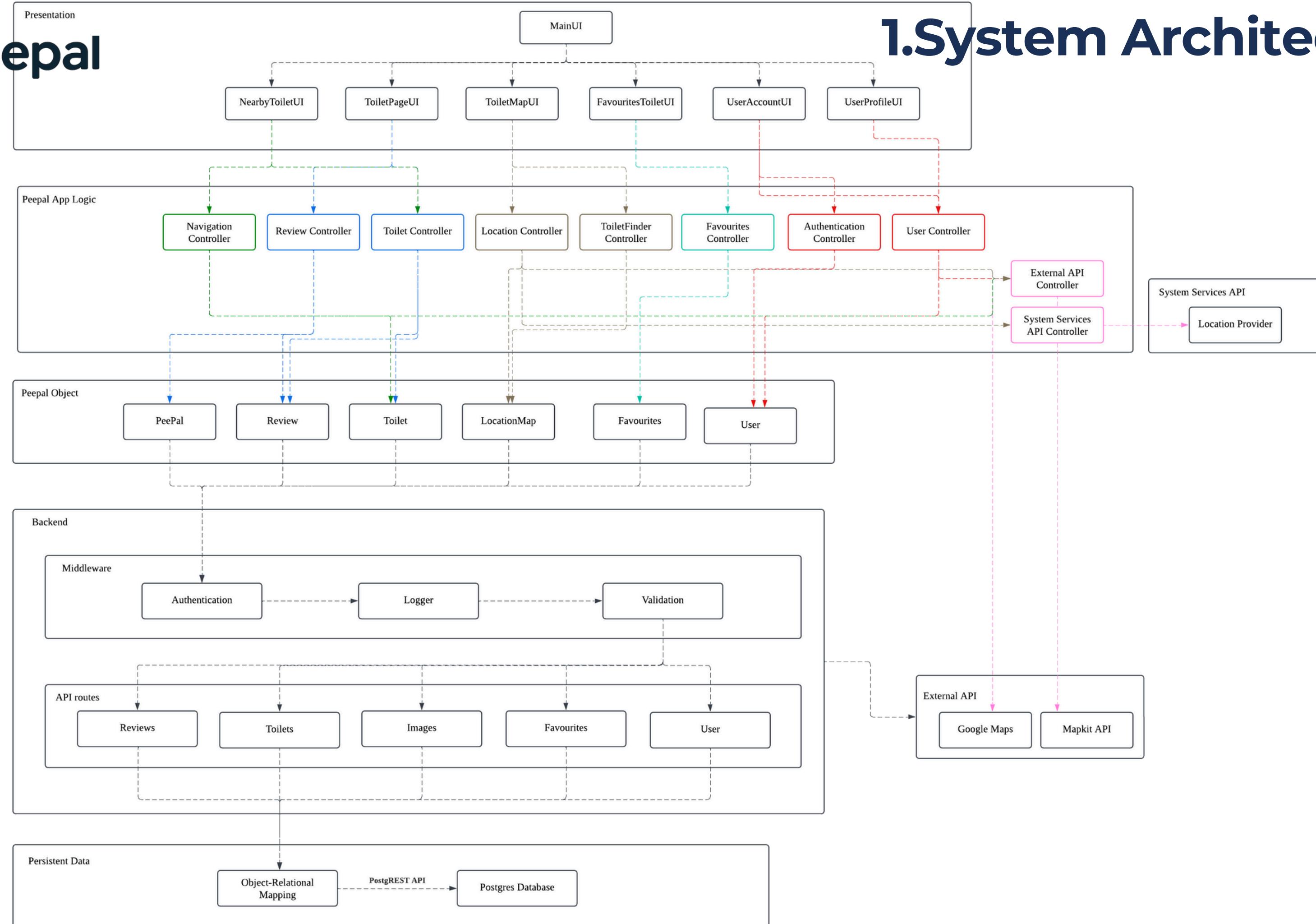
System Design



Presentation

peepal

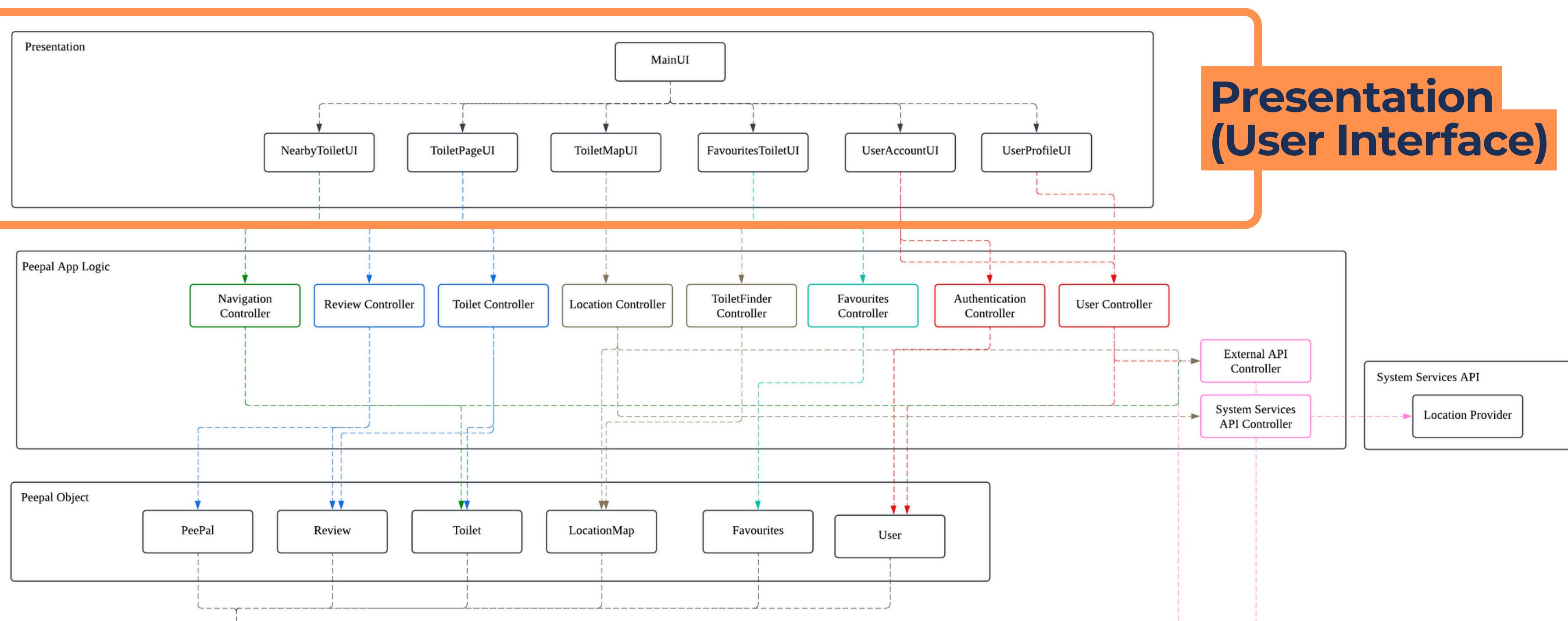
1. System Architecture





peepal

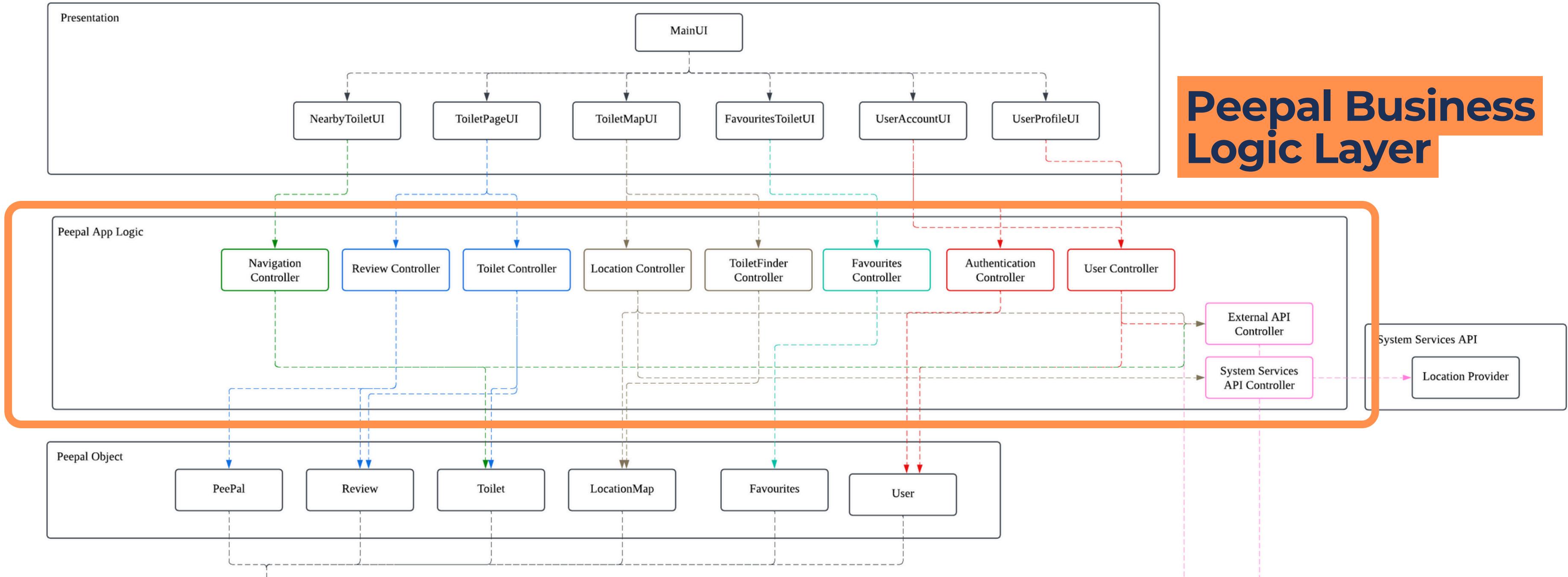
1. System Architecture





peepal

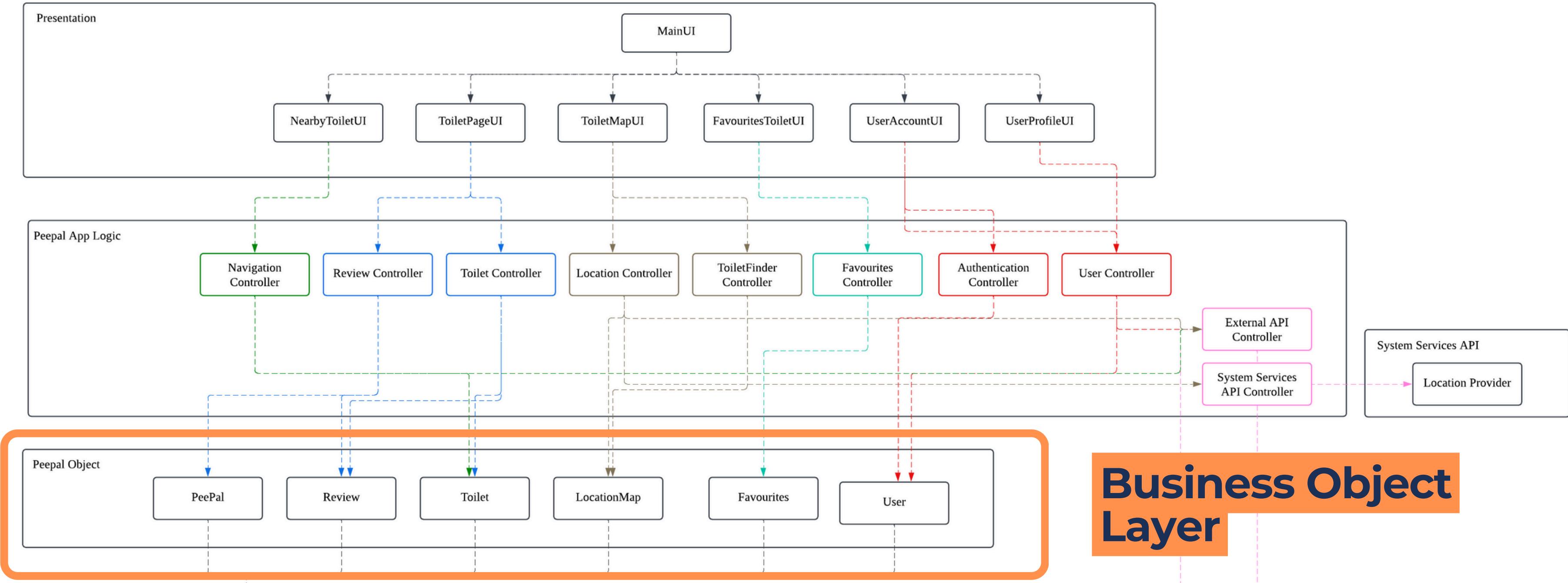
1. System Architecture





peepal

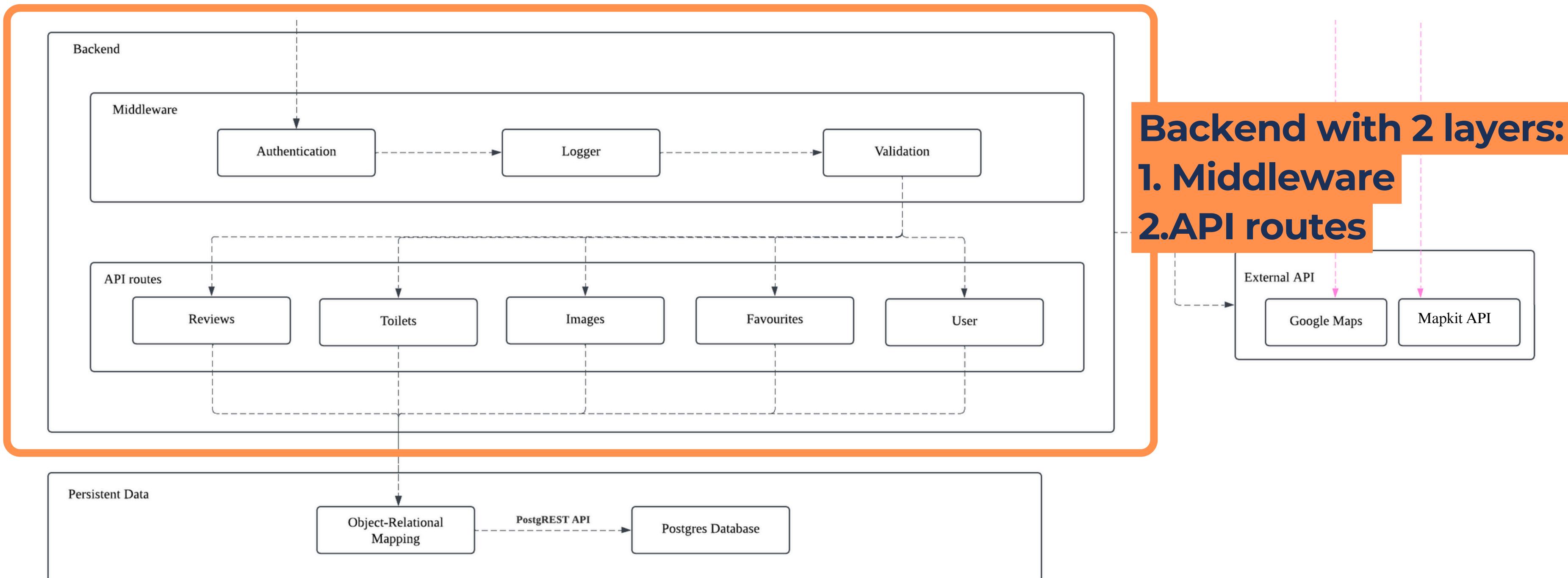
1. System Architecture





peepal

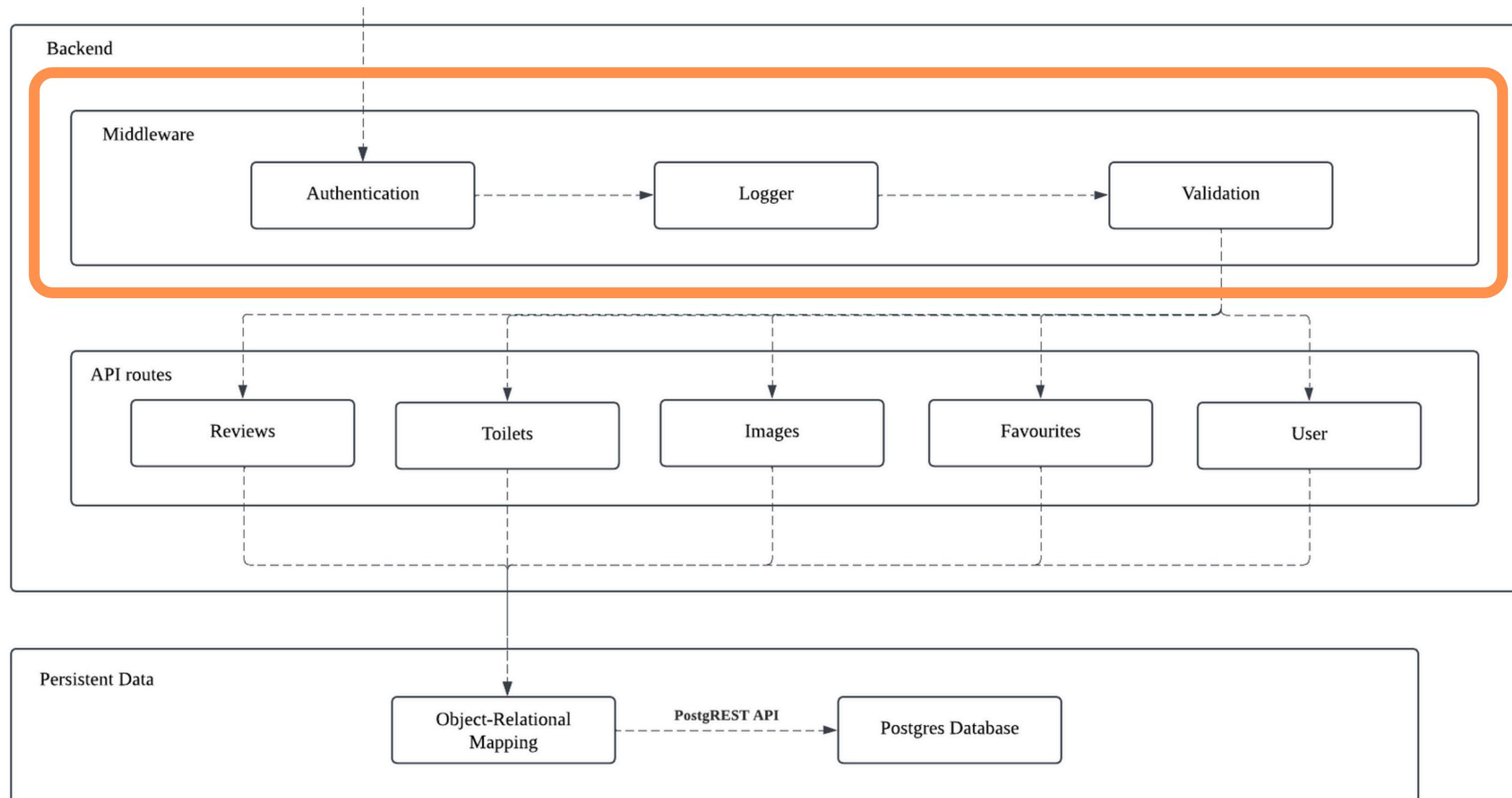
1. System Architecture





peepal

1. System Architecture

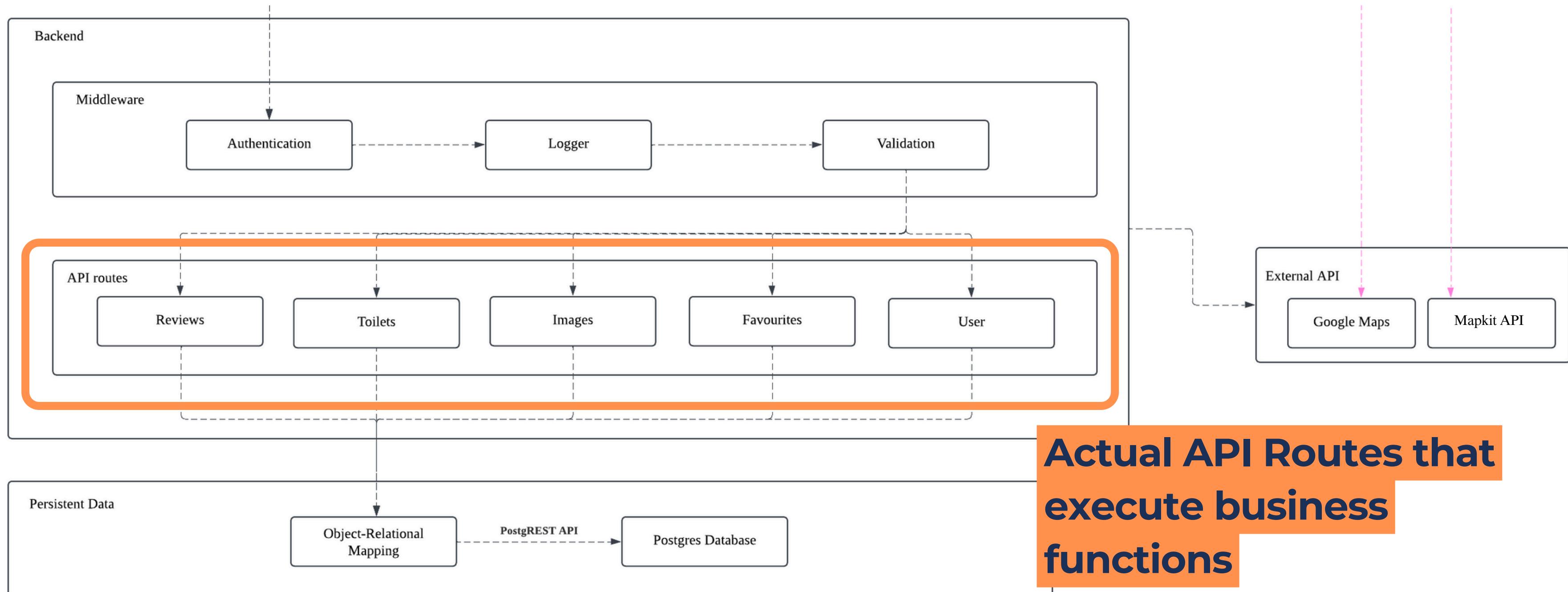


Middleware handle
1. Authentication
2. Logging
3. Data Validation



peepal

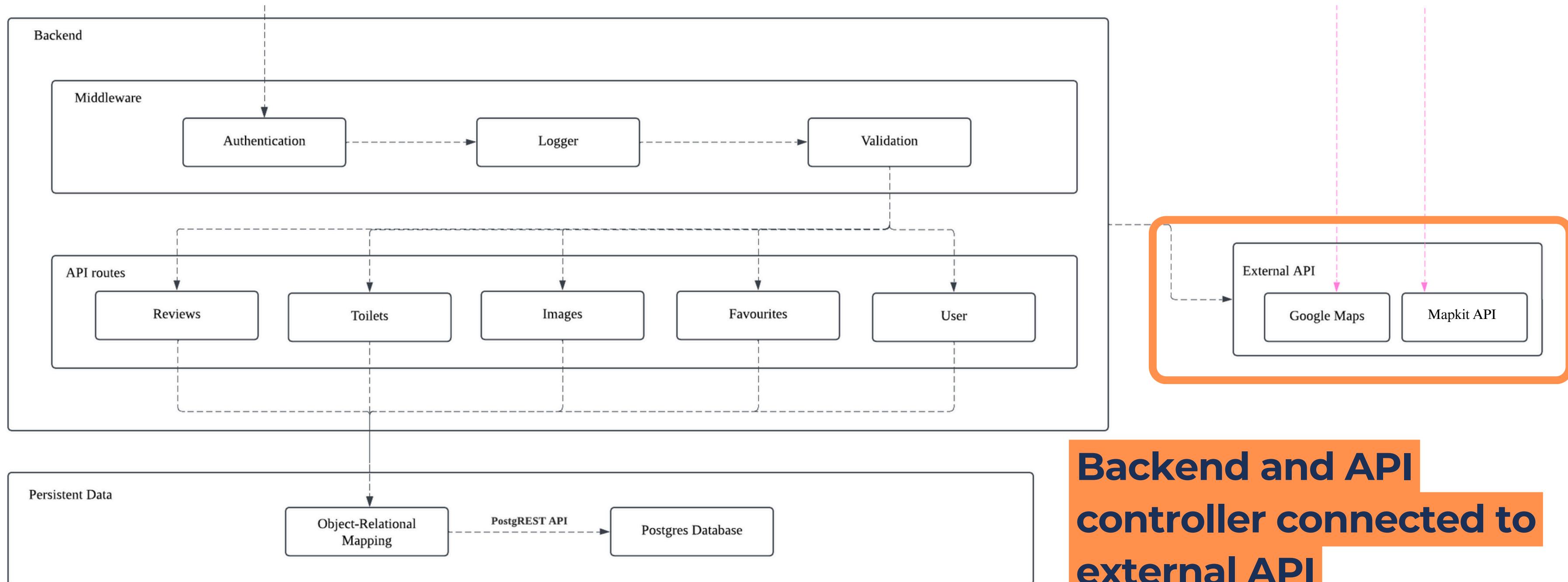
1. System Architecture





peepal

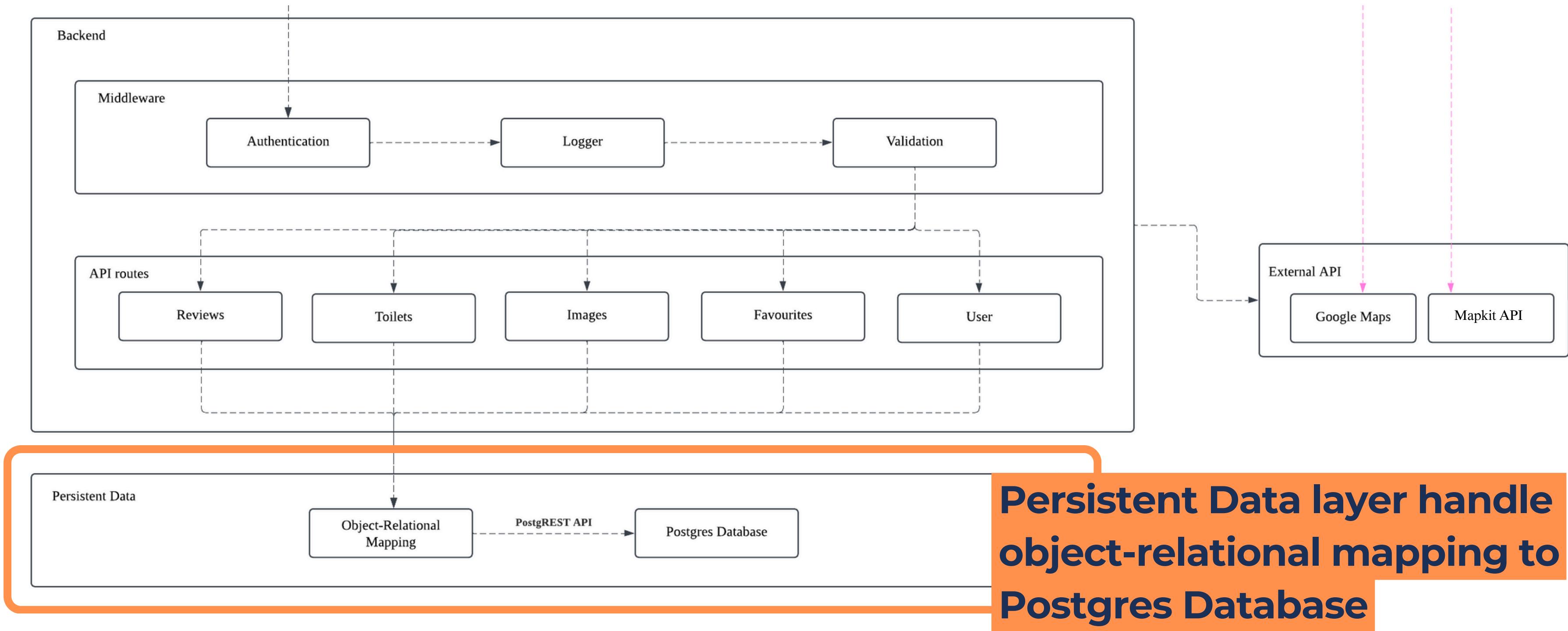
1. System Architecture





peepal

1. System Architecture





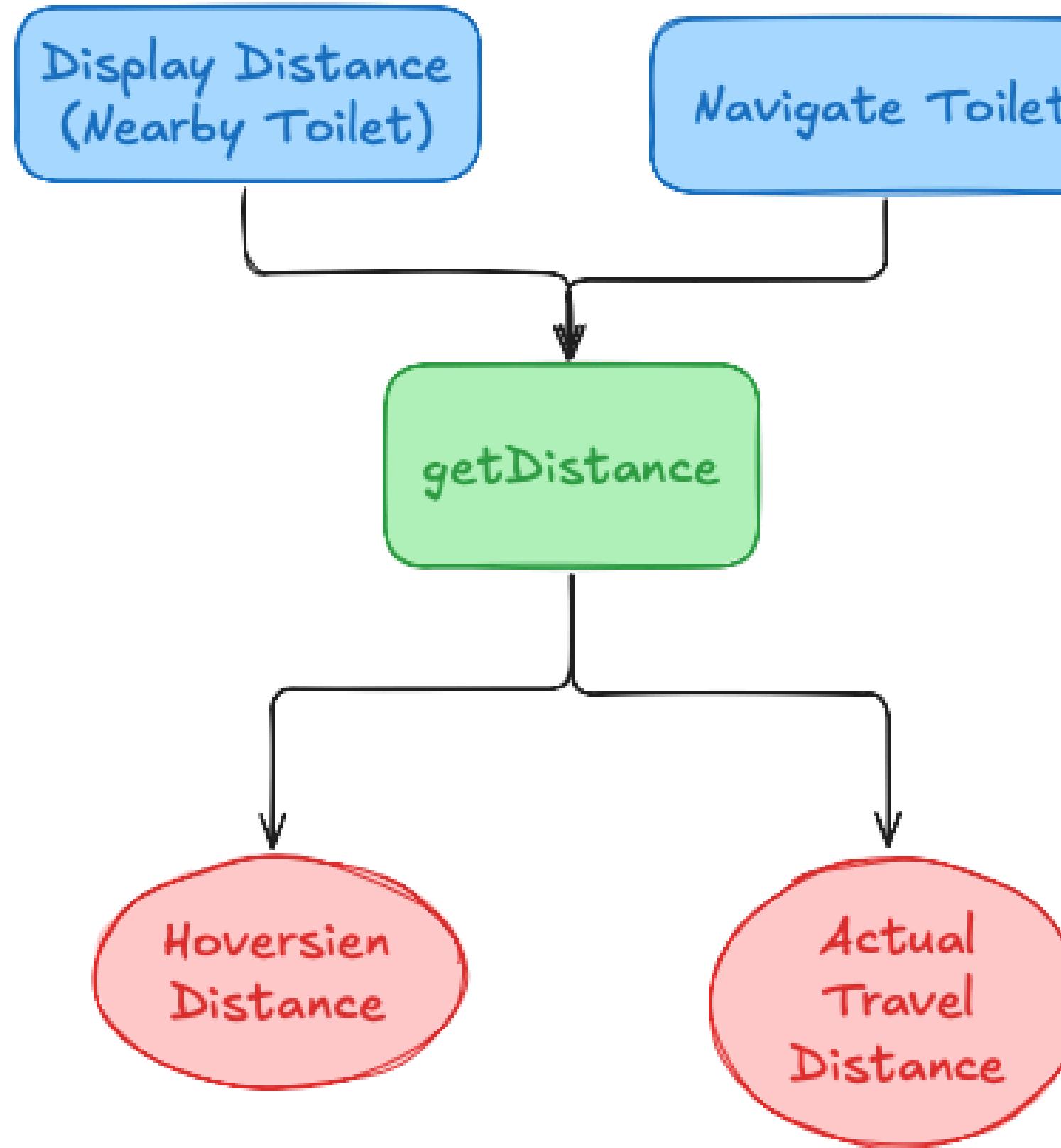
peepal



Design Patterns

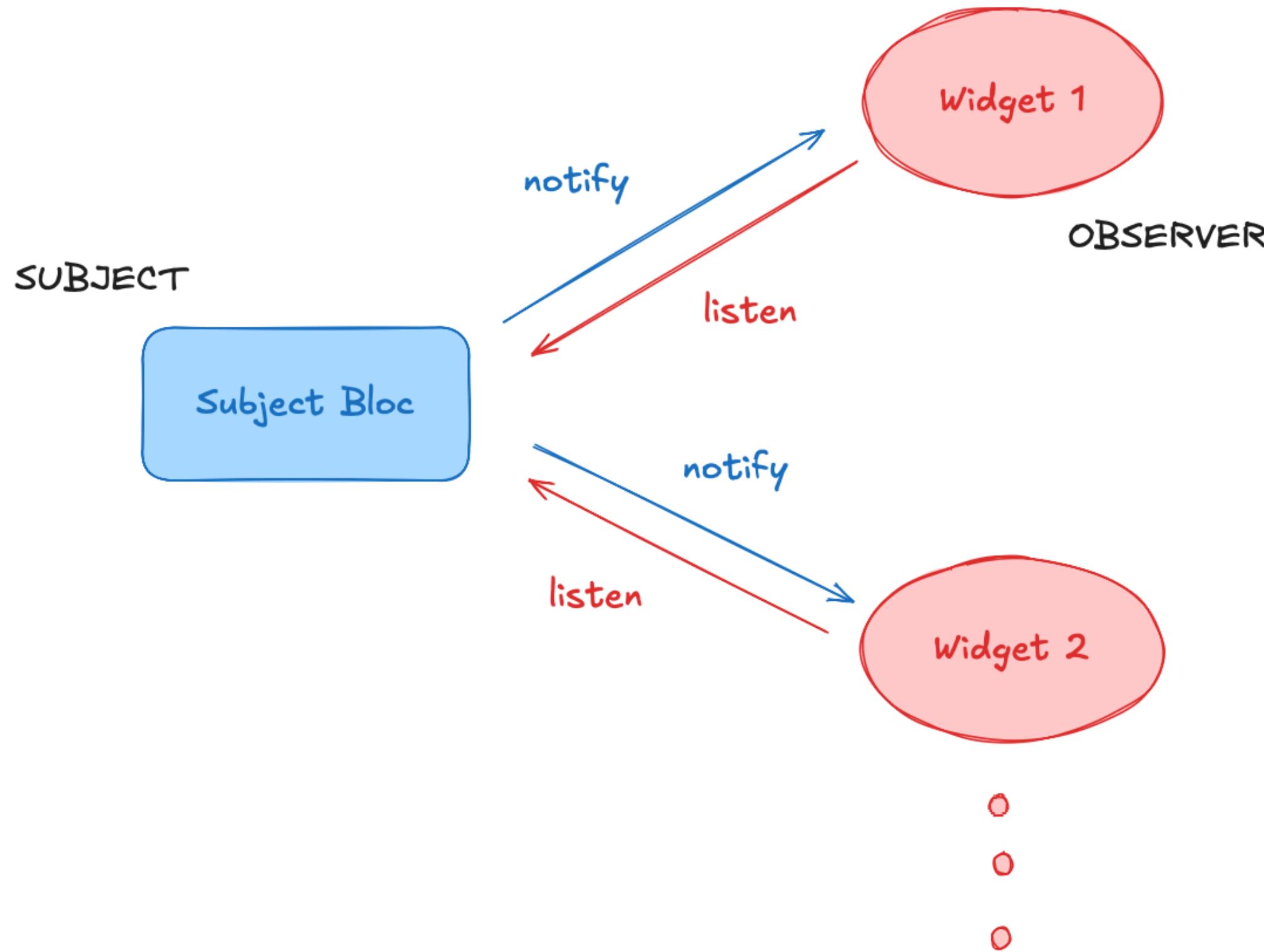


peepal



1. Strategy Pattern

- The system will get the location distance according to the widget specification.
- There are different implementations and calculation methods to get distance.

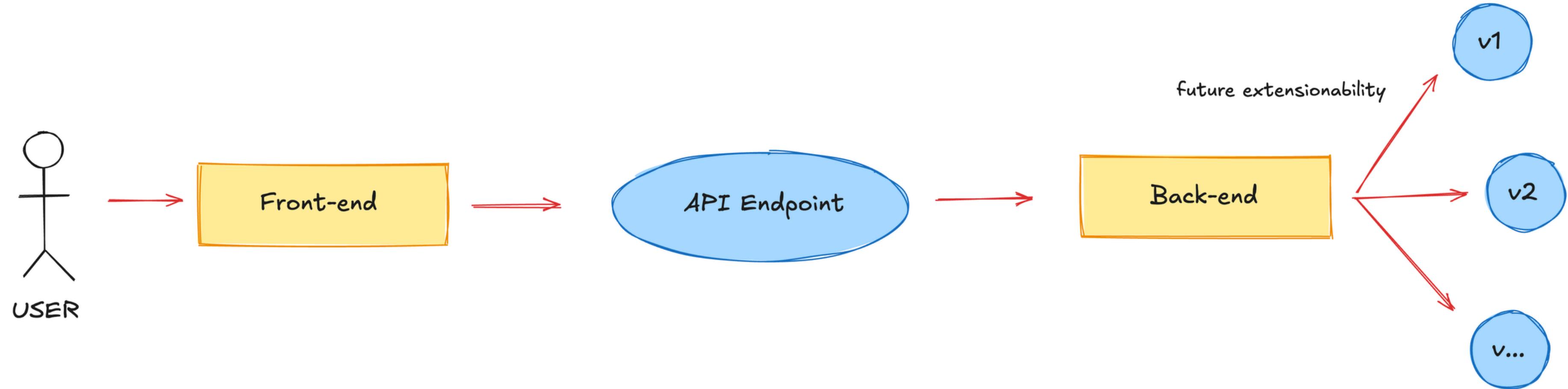


2. Observer Pattern

- UI components (Observers) subscribe to the bloc using **BlocBuilder** or **BlocListener**.
- When an event occurs, the bloc processes it.
- All subscribed widgets are automatically rebuilt with the state change.



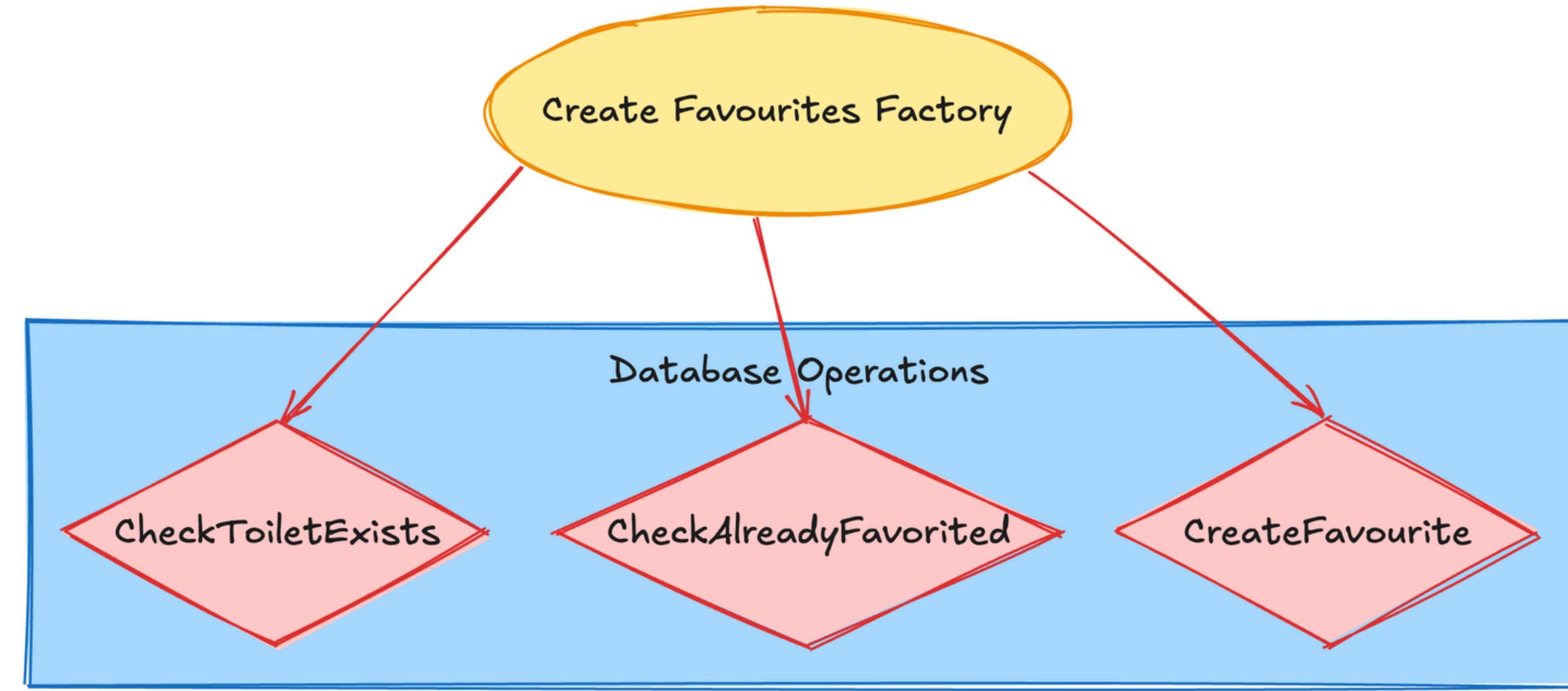
3. Facade Pattern



- When a client makes a POST request to /toilets/nearby, the Toilets API facade handles the request through handleGetNearby(). Behind the scenes, it coordinates multiple operations.
- The client only interacts with the simple Toilets API interface, unaware of the complex operations behind the scenes.
- This pattern improves code maintainability by encapsulating complexity and enhancing usability by providing a clean, simple interface for complex operations.



4. Factory Pattern



- The CreateFavoriteFactory manages favorite creation through the Factory Pattern.
- It first checks if the toilet exists, then verifies if it's already favorited, and finally creates the new favorite.
- This pattern hides complexity, maintains a single entry point, and makes modifications easy without affecting other parts of the system.



peepal

Traceability in Project Deliverables



peepal

NearbyToilet

Use-case Description

Use Case ID:	#4-7		
Use Case Name:	NearbyToilet		
Created By:	Jia Wei	Last Updated By:	Jia Wei
Date Created:	13/04/2025	Date Last Updated:	13/04/2025

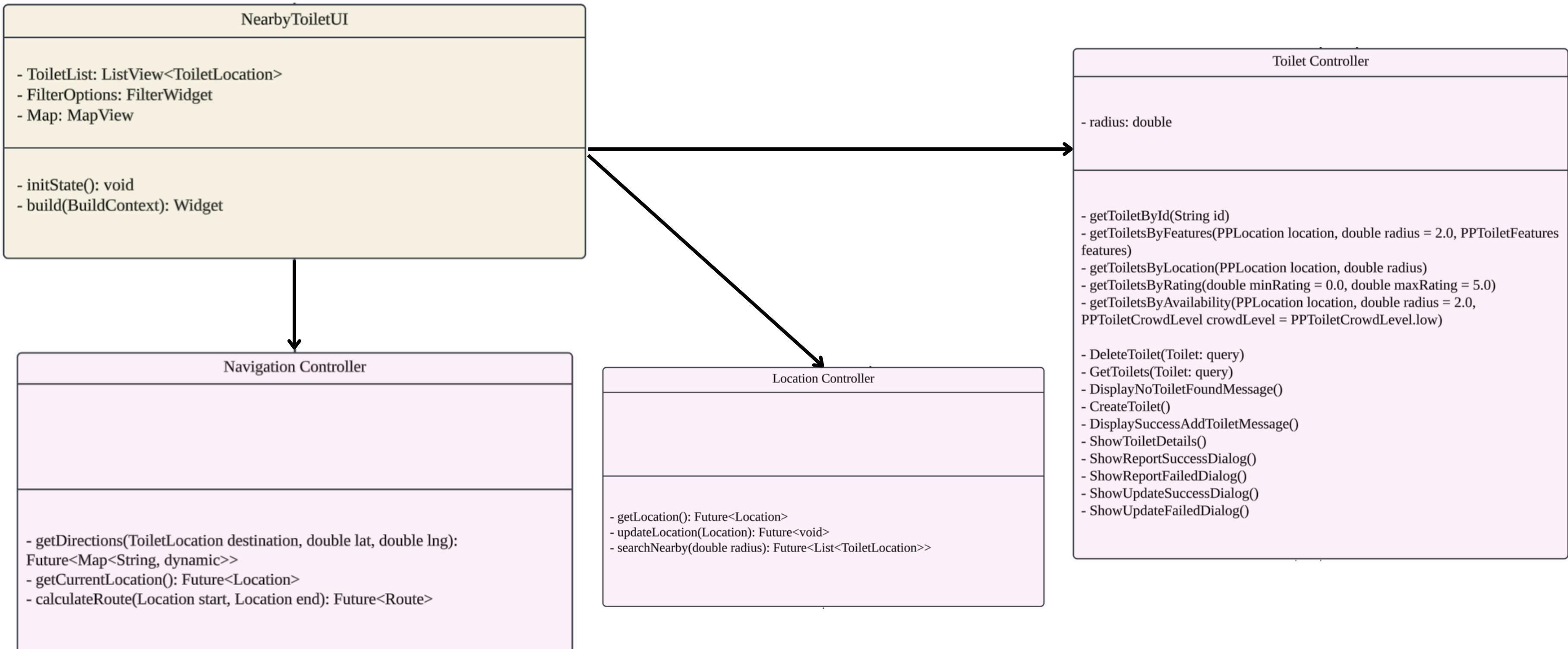
Actor:	User
Descriptions:	Display the information of the nearest 5 toilets in the card format.
Preconditions:	1. The User must be logged in and authenticated.
Postconditions:	1. Successfully display the nearby list of toilets.
Priority:	High
Frequency of Use:	High
Flow of Events:	<ol style="list-style-type: none">1. The system will detect the user's current location.2. The system will find the user's nearest 5 toilets.3. The system will display the information in the card format.4. The user can select any toilet card and the system will invoke ToiletMenu use case.5. If the user selects the "Navigate" button, the system will invoke ViewToiletOnMap use case.
Alternative Flows:	None
Exceptions:	None
Includes:	<ol style="list-style-type: none">1. ToiletMenu2. ViewToiletsOnMap
Special Requirements:	None
Assumptions:	None
Notes and Issues:	None



peepal

NearbyToilet

Relevant Class Diagram

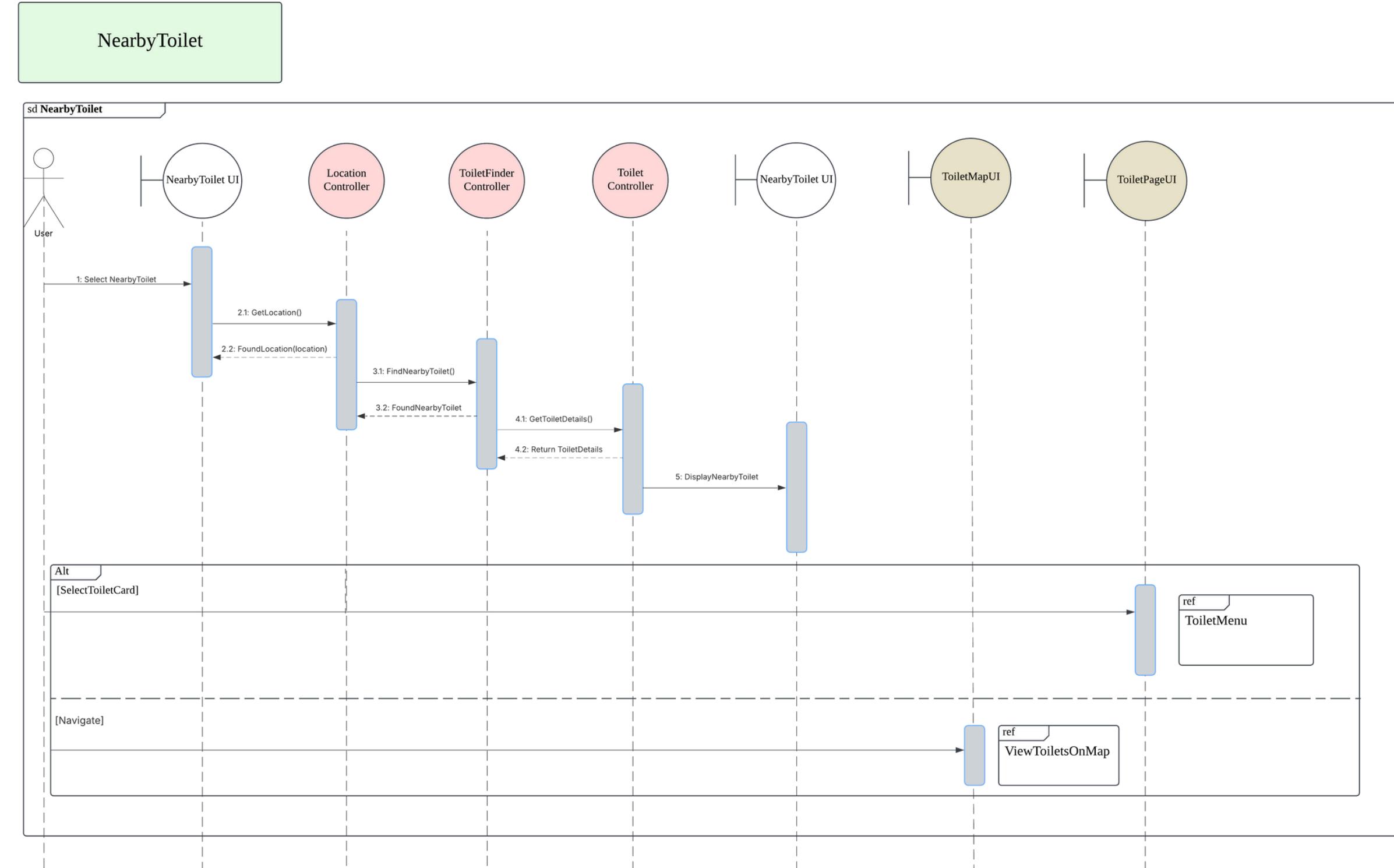




peepal

NearbyToilet

Relevant Sequence Diagram





peepal

NavigateToilet - Good Designs Applied

Facade Pattern

```
Backend / src / routes / api / ts / tomtst.ts > ...
356 validator('json', navigateToiletSchema), async (c) => {
357
358   const mkRoute: MKRoute = directionsResponse.data.routes[0]
359   const mkSteps: MKStep[] = directionsResponse.data.steps
360   const mkStepPaths: MKCoordinate[][] = directionsResponse.data.stepPaths
361
362   const stepPathTransformed: LatLng[][] = mkStepPaths.map(stepPath => stepPath.map(coord => ({ lat: coord.latitude, lng: coord.longitude })))
363   const stepPathPolylines: string[] = stepPathTransformed.map(path => encode(path))
364
365   const directions: RouteDirection[] = []
366
367   for (const step of directionsResponse.data.steps) {
368     const distanceString = formatDistance(step.distanceMeters)
369     const durationString = formatDuration(step.durationSeconds)
370     const polyline = stepPathTransformed[step.stepPathIndex]
371     const startLocation = stepPathTransformed[step.stepPathIndex][0]
372     const endLocation = stepPathTransformed[step.stepPathIndex][stepPathTransformed[step.stepPathIndex].length - 1]
373     const instructions = step.instructions
374
375     directions.push({
376       id: step.id,
377       distance: distanceString,
378       duration: durationString,
379       polyline: polyline,
380       start_location: startLocation,
381       end_location: endLocation,
382       instructions: instructions
383     })
384   }
385
386   const overviewPolylinePoints: {
387     x: number;
388     y: number;
389   }[] = stepPathTransformed.flat().map(coord => ({ x: coord.lng, y: coord.lat }))
390
391   const overviewPolyline: string = encode(simplify(overviewPolylinePoints, 0.0001, false).map(coord => [coord.y, coord.x]))
392
393   const distanceString: string = formatDistance(mkRoute.distanceMeters)
394   const durationString: string = formatDuration(mkRoute.durationSeconds)
395
396   const route: Route = {
397     overview_polyline: overviewPolyline,
398     start_location: {
399       lat: directionsResponse.data.origin.coordinate.latitude,
400       lng: directionsResponse.data.origin.coordinate.longitude
401     },
402     end_location: {
403       lat: directionsResponse.data.destination.coordinate.latitude,
404       lng: directionsResponse.data.destination.coordinate.longitude
405     },
406     distance: distanceString,
407     duration: durationString,
408     directions: directions
409   }
410
411   return c.json({ route: route }, 200);
412 }
```

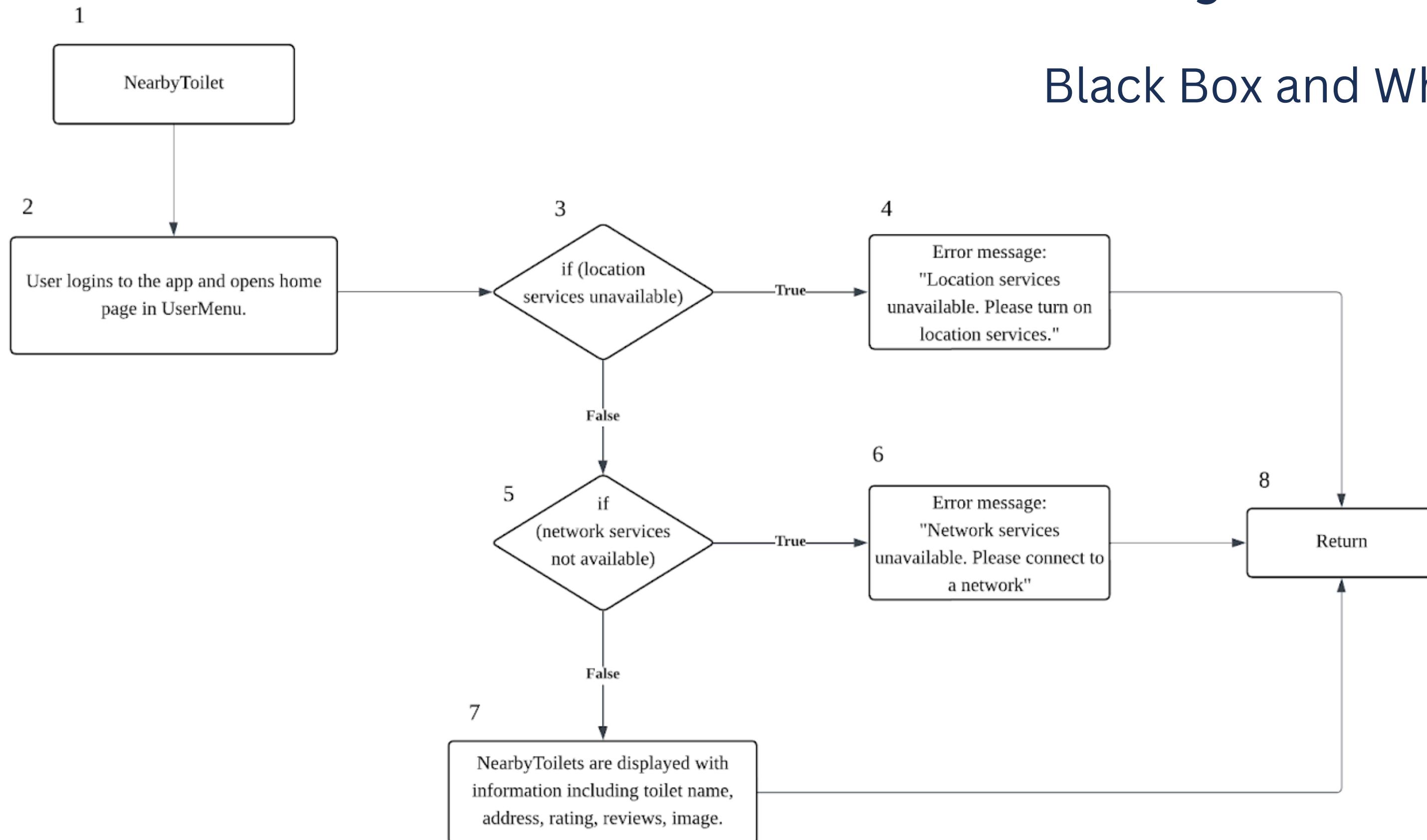
The single NavigateToilet API route handles many complex function calls:

1. Getting the MapKit **access token**
2. Getting the **directions**
3. **Parsing** the directions data
4. **Encoding** the data into polylines and routes that can be easily interpreted by the client.



peepal

NearbyToilet - Tests





Black Box and White Box

I.II Basic Path Testing

Cyclomatic Complexity = | decision points | + 1 = 2 + 1 = 3

Basis Paths

1. Baseline path: 1,2,3,5,7,8
2. Basis path 2: 1,2,3,4,8
3. Basis path 3: 1,2,3,5,6,8



Black Box and White Box

I.III Test Cases and Results

NearbyToilet

No.	Test Input	Expected Output	Actual Output	Pass?
1.	Location services available. Network services available.	Nearby Toilets displayed successfully.	Nearby Toilets displayed successfully.	Yes
2.	Location services unavailable. Network services available.	Error message: "Location services unavailable. Please turn on location services."	Error message: "Location services unavailable. Please turn on location services."	Yes
3.	Location services available. Network services unavailable.	Error message: "Network services unavailable. Please connect to a network"	Error message: "Network services unavailable. Please connect to a network"	Yes



peepal

Automated Unit Testing

```
Test Files  5 passed (5)
Tests      50 passed (50)
Start at   00:19:54
Duration   2.13s (transform 359ms, setup 0ms, collect 3.24s, tests 2.58s, environment 1ms, prepare 351ms)

PASS Waiting for file changes...
press h to show help, press q to quit
```



peepal

Continuous Integration/Deployment (CI/CD)

V peepal-backend-deployment Private

		Go to file	Add file	Code
main	1 Branch	0 Tags		
.vscode	deployment	2 hours ago		
drizzle	deployment	2 hours ago		
scripts	deployment	2 hours ago		
src	deployment	2 hours ago		
.dockerignore	deployment	2 hours ago		
.gitignore	deployment	2 hours ago		
README.Docker.md	deployment	2 hours ago		
README.md	deployment	2 hours ago		
compose.yaml	deployment	2 hours ago		
dockerfile	deployment	2 hours ago		
drizzle.config.ts	deployment	2 hours ago		
package-lock.json	deployment	2 hours ago		
package.json	deployment	2 hours ago		
tsconfig.json	deployment	2 hours ago		
vitest.config.ts	deployment	2 hours ago		

Deployments

History

Source	Trigger	Date & Deploy time
deployment	Automatic (on commit)	Apr 16, 2025, 6:49 AM 1m 0s
deployment	Automatic (on commit)	Apr 16, 2025, 6:34 AM 1m 9s
deployment	Manual (on Sevalla) Bryan Soong	Apr 16, 2025, 6:30 AM 21s
deployment	Automatic (on commit)	Apr 16, 2025, 6:27 AM 1m 53s
deployment	Automatic (on commit)	Apr 16, 2025, 6:09 AM 1m 6s
deployment	Manual (on Sevalla) Bryan Soong	Apr 16, 2025, 4:50 AM 22s
deployment		Apr 16, 2025, 4:45 AM

Deployment is in progress.

Cancel

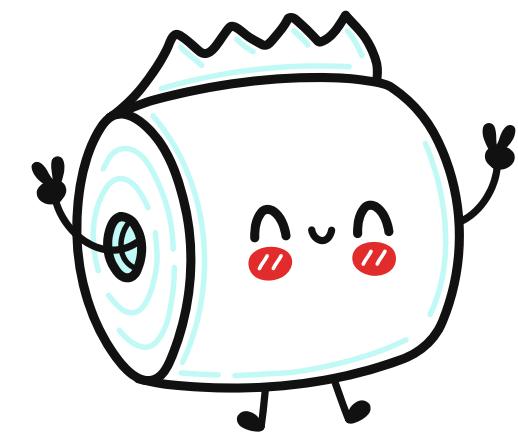


peepal

Future Plans



Health Notifications
Integrating official government health announcements and push notifications for users.



HappyToilet
Collaborating with the Restroom Association (Singapore) to feature data ratings from their Happy Toilet program.



Expand Globally
Transitioning from a Singapore-focused toilet database to a global platform.



peepal

Thank you!

Your personal toilet buddy.



peepal

Video Demo

Create Account

Create Account

Full Name

Email

Password

Gender

Create Account

i the i'm

q w e r t y u i o p

a s d f g h j k l

z x c v b n m

123 space done