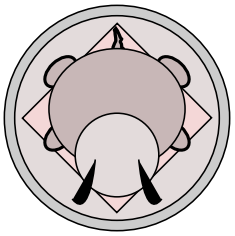


Project Specifications of Blooper App

(Version: 1.5)

Whoopie Bard <whoopie@clueless.dev>
Changu Bhai <changu.bhai@clueless.dev>
14 June, 2025

Contents

1 Document Info

1.1 Authors

Authors	Email	Nickname
Changu Bhai	changu.bhai@clueless.dev	changu
Whoopie Bard	whoopie@clueless.dev	whoopie

1.2 Document Versions

Versions	Author	Changes	Date
v 1.0	whoopie	Added introductory text	2 October, 2024
v 1.1	changu	Completed functional specifications	28 October, 2024
v 1.2	whoopie	Updated partial program specifications	15 November, 2024
v 1.3	whoopie	Added DB schema	24 December, 2024
v 1.4	changu	Added section Track Changes	9 February, 2025
v 1.5	whoopie	Added use-cases	14 June, 2025

1.3 Track Changes

No.	*Type*	*Changes*
1	✖ CUT	No two user-accounts should be possible to be created using same mobile number.
2	+ NEW	Multiple user-accounts should be able to be created using the same mobile num...
3	✖ CUT	Subsubsection 3.1.3
4	✖ CUT	List Item 3.1.4.3.2.1
5	+ NEW	List Item 3.1.4.3.2.4

6	✖ CUT	{
		”status”: 421,
		”message”: ”Field supplied is not valid”
		}

7	+ NEW	{
		”status”: 422,
		”message”: ”Invalid field supplied”
		}

2 Introduction

Clueless Developers' Consortium has been allocated the project to develop Bloopers App. This document defines the specifications of that project. It is divided primarily into two parts - Functional Specifications and Program Specifications. For any queries regarding this project, please contact whoopie@clueless.dev.

It is sincerely hoped that this project specifications document would be found extremely useful by the client to manage and maintain the Blooper App properly for a very long period of time.

3 Functional Specifications

This section provides the detailed functional specifications of Blooper Application.

3.1 User Registration

This module defines the user-registration aspect of the Blooper. Anonymous users are allowed to self-register and proceed to use the application.

↰ User-registration Module

3.1.1 Requirement Specifications

Following requrements specifications are applicable to the user-registration module of the application.

1. New user should be able to self-register to the Blooper application. Using a specified URL, anonymous users would be able to fetch a web-form to fill-up.
2. Users with an email address and mobile number should be able to create an account in Blooper.
3. No two user-accounts should be possible to be created using same email address.
4. CUT ✂ No two user-accounts should be possible to be created using same mobile number.

↰ Track Changes: 1

5. NEW ✚ Multiple user-accounts should be able to be created using the same mobile number.

↰ Track Changes: 2

6. User should be able to register from certain geographical regions only.

   Q: How to prevent people from using VPN to beat geographical restrictions?
   A: It is not possible to do.
   Q: Is there any third party service, which provides such facilities?
   A: Not known.

3.1.2 Wireframes

Following wireframes are proposed for the user-registration screen.

3.1.2.1 User-Registration Form

This screen is displayed to a new user whose user-account is yet to be created. Initially, all fields turn out empty. The user needs to fill the required fields in this form.

↰ Wireframes for User Registrat...
↰ User-registration Screen
↰ Login

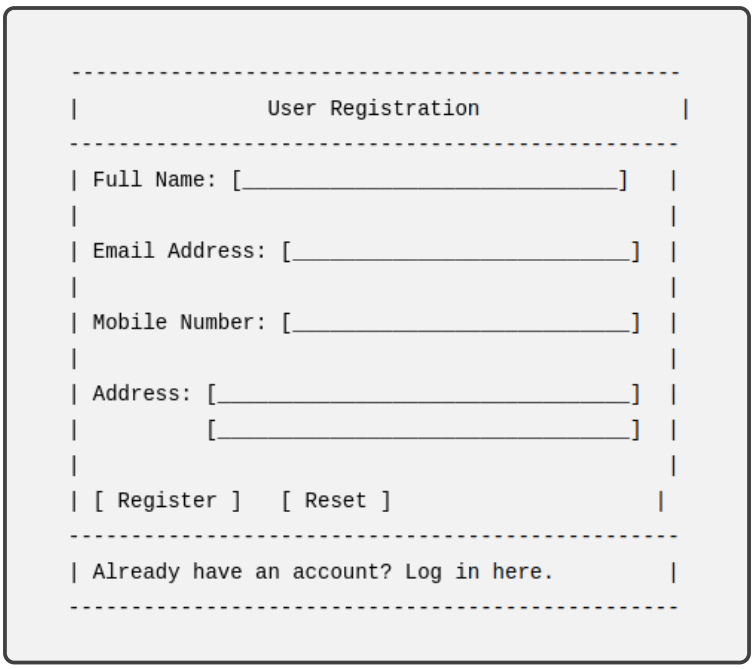


Figure 1: User-registration Form

3.1.2.1.1 Form-Fields

Following fields are expected to be present in this form.

Form-Fields	Allowed Values	Max Size	Required	Unique
Address	A-Z, a-z, 0-9, - +	char(512)	No	
Email Address	.a-zo-9-_+@	char(128)	Yes	Yes
Full Name	A to Z, and a to z	char(128)	No	
Mobile Number	+-0123456789	char(12)	Yes	Yes

- **Email Address**:
  - It would be unique across the user-database.
  - That means, no two users can have same email address.
- **Mobile Number**:
  - Mobile number should be unique across the application-database.
  - No two users using this application can have same mobile number.

3.1.2.1.2 Buttons

Following buttons are expected to be made available on the screen.

1. **Register**: Registers the new user.
   (a) **Normal Flow**

i. System validates supplied data before creating user-account.
ii. If everything is in order, then system create new user-account.

(b) **Exception Flow**
If one or more validations fail, then system displays validation error with following details.

i. If email address is associated with any existing user-account, then system notifies that no new account can be created with same email address.
ii. If mobile number is associated with any existing user-account, then system notifies the user that no new account can be created with the same mobile number.

2. **Reset**: Reset all fields of the registration form.

(a) **Normal Flow**

i. Clean-up contents of all fields.

↰ Reset

3.1.2.1.3 Links

Following hyperlinks are expected to be present on the screen.

- **Login Here**: If account is present, click to sign-in.

3.1.2.2 Warning Message for Duplicate User-Account

This screen is displayed, when one or more fields which are unique across the application is supplied as part of user-registration, but the same had been supplied earlier for another user-account too.
This could be a pop-up window.

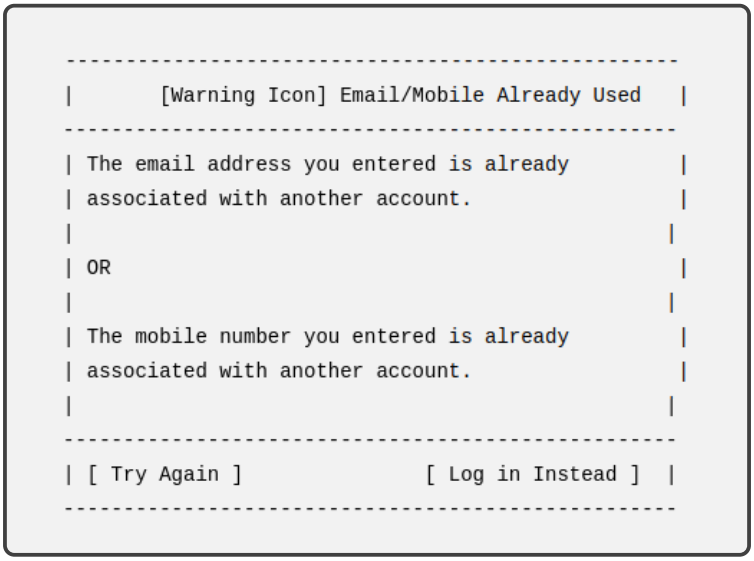


Figure 2: Message Box for Existing Record

↰ Error-Message Box for Existin...

3.1.2.2.1 Buttons

1. **Try Again**: Re-submit the form by going to the registration form and by modifying its form-fields.

   Q: Is re-executing the GET request on the form required here?
   Q: Is it not sufficient enough to handle it locally, by rendering the cached form itself?

2. **Login Instead**: If an account exists already, proceed to the login form.

3.1.3 CUT Subsubsection 3.1.3

Q: How can Subsubsection 3.1.3 be used as a section heading?

↰ Track Changes: 3

Sample Table	Header 1	Header 2	Header 3	Header 4
Row 1	1.1	1.2	1.3	1.4
Row 2	2.1	2.2	2.3	2.4
Row 3	3.1	3.2	3.3	3.4
Row 4	4.1	4.2	4.3	4.4
Row 5	5.1	5.2	5.3	5.4

- **Row 1**:
  - First note for Row 1.
  - Second note for Row1.
  - Third note for Row1.
- **Row 3**:
  - First note for Row 3.
  - Second note for Row 3.
  - Third note for Row 3.
  - Fourth note for Row 3.

3.1.4 Subsubsection 3.1.4

1. List Item 3.1.4.1
2. List Item 3.1.4.2
   (a) List Item 3.1.4.2.1
   (b) List Item 3.1.4.2.2
   (c) List Item 3.1.4.2.3
   (d) List Item 3.1.4.2.4
   (e) List Item 3.1.4.2.5
3. List Item 3.1.4.3

(a) List Item 3.1.4.3.1

(b) List Item 3.1.4.3.2

i. CUT List Item 3.1.4.3.2.1 ↰ Track Changes: 4

ii. List Item 3.1.4.3.2.2

iii. List Item 3.1.4.3.2.3

iv. NEW List Item 3.1.4.3.2.4 ↰ Track Changes: 5

v. List Item 3.1.4.3.2.5

vi. List Item 3.1.4.3.2.6

(c) List Item 3.1.4.3.3

> Be warned!!!

(d) List Item 3.1.4.3.4

i. List Item 3.1.4.3.4.1

ii. List Item 3.1.4.3.4.2

iii. List Item 3.1.4.3.4.3

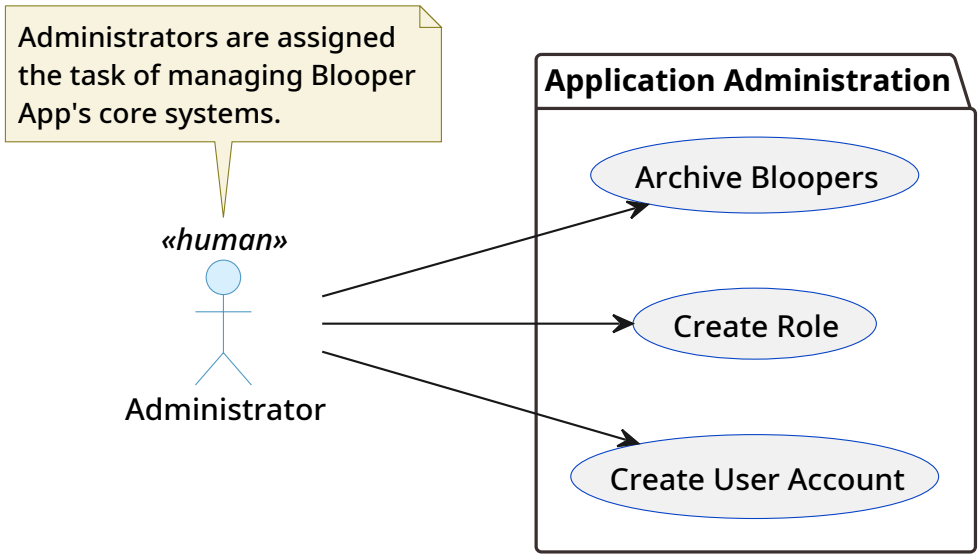
iv. List Item 3.1.4.3.4.4

3.2 Use Cases

Following use-cases are applicable for the Blooper App.

3.2.1 Application Administration

This package contains usecases pertaining to the administrative activities of the application.



Create User Account

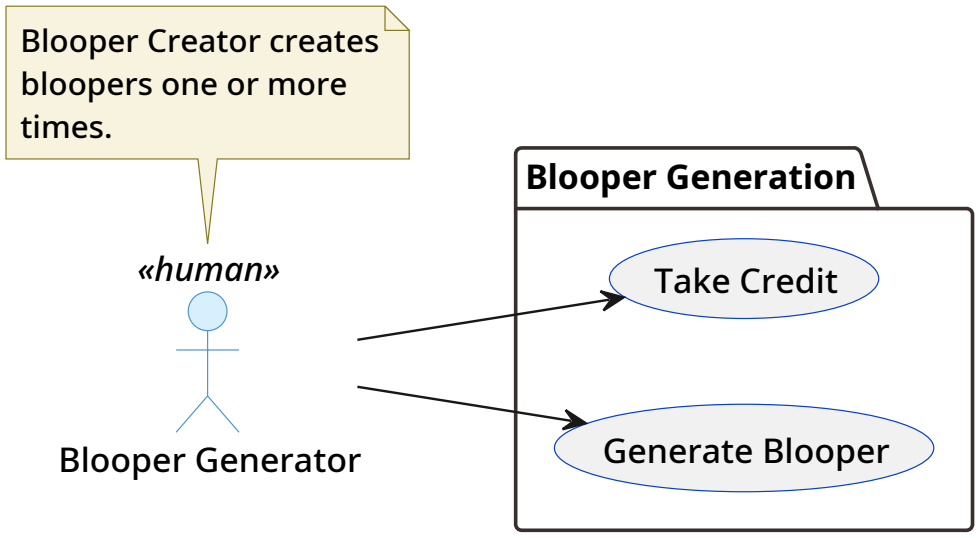
Create User Account	
Normal Flow	System provides a form to enter details of user-account.
	User enters mandatory and optional data into the form.
	User submits the form.
	System persists the data in a database.

Create Role	
Normal Flow	System provides a form to enter details of user-roles.
	User enters mandatory and optional data into the form.
	User submits the form.
	System persists the data in a database.

Archive Bloopers	
Preconditions	One or more bloopers with remedial actions must exist.
Normal Flow	System lists down all bloopers having respective remedial actions taken.
	User selects one or more bloopers from list.
	User submits selected bloopers for archival.
	System moves the selected bloopers to the archive.
	System removes the selected bloopers from the list.
Alternate Flow	If one or more selected bloopers are marked for not archival, do not list them in archival screen.
Exception Flow	In case moving blooper to archive fails, do not remove it from the blooper list.

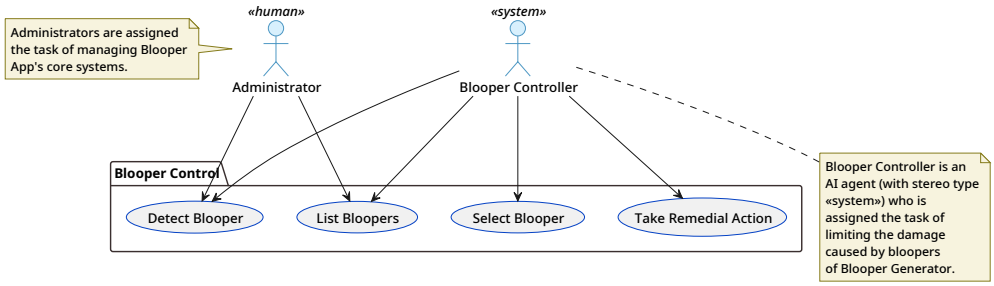
3.2.2 Generation of Bloopers

This package defines usecases involved in generation of bloopers.



3.2.3 Blooper Control

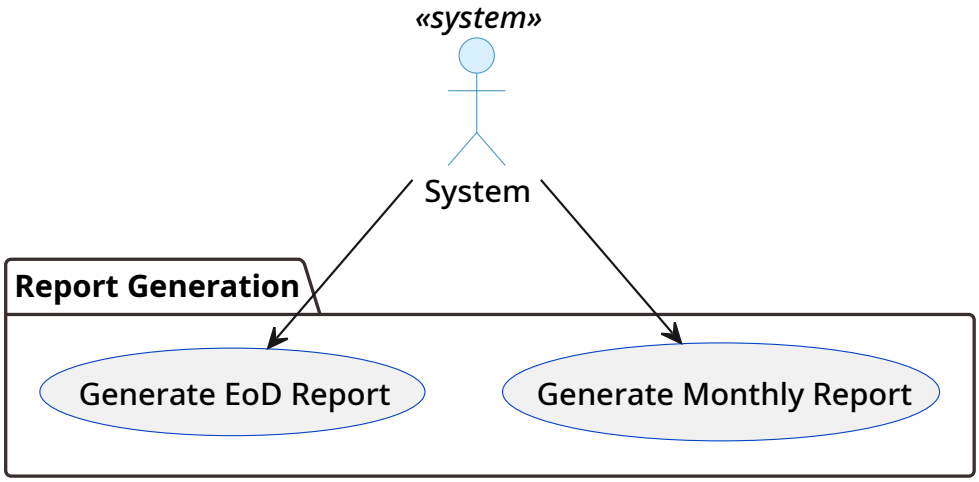
Following are the usecases pertaining to controlling bloopers.



Select Blooper	
Preconditions	One or more bloopers must exist already.
Normal Flow	Search criteria is supplied by user to search bloopers based on them.
	System returns a list of matching bloopers which is rendered in a table with pagination enabled.
Alternate Flow	System returns an empty list to indicate that no bloopers matched supplied search criteria.
Exception Flow	User doesn't have rights to search and list bloopers.

3.2.4 MIS

All report-related usecases are collected here.



4 Effort Estimations

The estimation of work is calculated using function-point analysis method. It was reviewed by following people:
1. Mr. Changu Bhai

2. Mrs. Mai In Jun

Following table summarizes the approximate efforts (in person-hours) required for designing, developing, preparation of unit tests, functional testing, and documentation for each module of this application.

Modules	Des.	Dev.	U. Test.	F. Test.	Doc.	Remarks
User Registration	2	9	6	2	1	
Password Reset	3	4	4	6	2	
Blooper Generation	4	10	8	2	2	Complex module
Total	**9**	**23**	**18**	**10**	**5**	

5 Risks

Following risks are identified for this project.

1. Developer Attrition

Risk-ID: RSK-01

Risk-Types: Time, Financial

- It is possible that skilled developers leave the project mid-way of execution.
- As a remedy, a pool of skilled resources should be kept on bench, which might increase the financial risk.

2. Delay in Communication with Client

Risk-ID: RSK-02

Risk-Types: Time

- The delay in receiving clarifications from client on critical matters would affect the timelines of delivery.
- The turnaround time for such communication threads should be well-defined and included in the contract-terms to mitigate this risk.

3. Usage of unsupported Open Source solutions

Risk-ID: RSK-03

Risk-Types: Compliance

- The data-cleaning solution used in this application is not maintained by the open source community anymore. Its last update was found to be four years old.
- As a mitigation strategy, funding should be provided to revive and maintain this open source project and release the new version with fixes for reported bugs.

6 Modules and Deliverables

Following modules and deliverables are identified for phase 1 of this project.

Deliverable-ID: DLV-01

Deliverable	LDAP based User Authentication Module
Accountability	Mohan Kumar
Date of Delivery	November 30, 2025

1. Implementation of module for user-authentication using LDAP.
2. An extension library named blooper_ldap_ext which will allow other applications to use Blooper App as their SSO service provider by using its own LDAP backend.

Deliverable-ID: DLV-02

Deliverable	Program Specification for SSO Integration
Accountability	Mai In Jun
Date of Delivery	November 08, 2025

Deliverable-ID: ~~DLV-03~~

Deliverable	~~Password Reset Form~~
Accountability	Jojo Dey
Date of Delivery	November 12, 2025

Deliverable-ID: DLV-04

Deliverable	Wireframes for User Authentication UI
Accountability	Uday Khedekar
Date of Delivery	November 22, 2025

Deliverable-ID: DLV-05

Deliverable	Wireframes for User Registration UI
Accountability	Shabnam Ansari, Ramji D'souza
Date of Delivery	November 16, 2025

Screen mockups for user registration module is to be developed. The wireframes for the same is available at User-registration Form.

Deliverable-ID: DLV-06

Deliverable	Realm-restriction Middleware
Accountability	Josephine Walters
Date of Delivery	November 19, 2025

A middleware to ensure that incoming requests from the authenticated users do not cross the realm-boundary. The realm-code applicable for that user would be injected automatically into the request.

Deliverable-ID: DLV-07

Deliverable	Email based Password Reset Module
Accountability	Uday Khedekar
Date of Delivery	November 16, 2025

Users should receive an email containing link to reset the password, subject to additional confirmation of date of birth, or some other secret questions set by the user.

7 Program Specifications

7.1 User-registration Module

This section elaborates on the program specifications of implementation of user-registration module of Blooper.
The functional requirements of this module is described in User Registration.

7.1.1 Screens

Following screens are developed for user-registration purpose.

7.1.1.1 User-registration Screen

Please refer User-registration Form for wireframe of user-registration screen.

7.1.1.1.1 Buttons

- **Register**

  The Normal Flow provides the details of the normal flow of actions when this button is clicked.

  The Exception Flow provides the details of the exception flow of actions when this button gets clicked.

  - **API Endpoint**: /api/v1/users/register
  - **Method**: POST
    HTTP POST method is used on this API endpoint for the intended result.
    POST method requires input data to be sent in HTTP request-body itself.

- **Input**
  Input formats are described below.
  * {
    "fullname": "Jagmeet Chautala",
    "emailaddress": "jagmeetc@example.com",
    "mobilenumber": "1234567890",
    "address": "72, B street, Park Avenue, Sonpat,
    Hryana, INDIA"
    }
    This is the JSON format of input.
  * {
    "fullname": "Jagmeet Chautala",
    "emailaddress": "jagmeetc@example.com",
    "mobilenumber": "1234567890"
    }
    It is also possible that some partial data is supplied by ignoring optional details.
- **Normal Output**
  * {
    "status": 200
    "message": "Created"
    }
- **Exception Output**
  * CUT
    {
    "status": 421,
    "message": "Field supplied is not valid"
    }
    ↰ Track Changes: 6
  * NEW
    {
    "status": 422,
    "message": "Invalid field supplied"
    }
    ↰ Track Changes: 7

- **Reset**
  On-click, it empties all the fields of the form as also explained in Reset: Reset all fields of th....
  - **Method**: Javascript function to reset all form-fields.

7.1.1.1.2 Links

- **Login**
  This hyperlink takes the user to the login page as shown in User-registration Form.
- **CDC**
  This hyperlink takes the user to an external site, which is official website of the company Clueless Developers' Consortium.

7.1.1.2 Error-Message Box for Existing Record

Message Box for Existing Record shows the wireframe containing the error message described in this section.

7.1.1.2.1 Buttons

1. **Try Again**: Clicking on this button takes the user to the pre-populated user-registration page.
2. **Login Instead**: Clicking on this button takes the user to the login page.

7.1.2 Unordered List

7.2 Subsection 7.2

7.2.1 Subsubsection 7.2.1

7.2.2 Subsubsection 7.2.2

7.2.3 Subsubsection 7.2.3

7.2.3.1 Subsubsubsection 7.2.3.1

7.2.3.2 Subsubsubsection 7.2.3.2

8 Database Schema

In this section, we describe the database schema of databases used in Blooper App.

8.1 Relational Databases

Blooper App would be using HisSQL as its RDBMS in the backend. Multiple tables are required to implement the features expected from the this application. An RDBMS is required to maintain the integrity as well as consistency of the data-records pertaining to this application.
Following details are pertinent to the RDBS schema used in Blooper App.

1. users

↰ org_admins: user_id
↰ bloopers: blooped_by
↰ bloopers: detected_by

field	*type*	*unique*	*null*	*default*
`id` 🔑↑	int			
`name`	varchar[64]	yes	no	
`email`	varchar[80]			null
`contact_no`	varchar[16]			null
`dob`	date		no	
`status`	char[1]		no	

- `email`: Only one email address is allowed in this field.
- `dob`: Date in ISO date format.
- `status`: (A)ctive, (I)nactive, (D)eleted

2. organizations

field	type	unique	null	default
id ↑	int			
name	varchar[64]	yes	no	
address_id (addresses)	int		no	
status	char[1]		no	A

↰ org_admins: org_id
↰ bloopers: affected_by

- `status`: (A)ctive, (I)nactive, (D)eleted

3. org_admins

field	type	unique	null	default
id ↑	int			
org_id (organizations)	int		no	
user_id (users)	int		no	
since	date		no	

- `since`: The date from which this user is assigned the role of administrator for this organization.

- One organization may have one or more administrators. They are mapped using this table.
- The "since" field indicates a date when a particular administrator was mapped to an organization.

4. addresses

field	type	unique	null	default
id ↑	int			
address1	varchar[128]		no	
address2	varchar[128]		yes	
location_id (locations)	int		no	

↰ organizations: address_id

5. locations

field	type	unique	null	default
id ↑	int			
name	varchar[64]		no	
state_id (states)	int		no	

↰ addresses: location_id

- Unique(name, state_id)

6. states

↰ locations: state_id

field	type	unique	null	default
id 🔑↑	int			
name	varchar[64]		no	
code	varchar[8]		yes	
country_id (🔑 countries)	int		no	

- Unique(country_id, code)
- Unique(country_id, name)

7. countries

↰ states: country_id

field	type	unique	null	default
id 🔑↑	int			
code	varchar[4]	yes	no	
name	varchar[64]	yes	no	

8. bloopers

field	type	unique	null	default
id 🔑↑	int			
blooped_by (🔑 users)	int		no	
loss_amount	decimal[10,2]			
blooped_at	date		no	
detected_by (🔑 users)	int		no	
affected_by (🔑 organizations)	int		no	
meta_info	json		yes	

- `loss_amount`:
  - The money (in INR) lost due to this blooper.
  - Up to two decimal values, it is captured.

  `detected_by`: The user who detected this blooper.
- `affected_by`: The organization affected by this blooper.
- `meta_info`: Any additional information pertaining to this blooper would be stored here. There is no fixed schema for it. The object using this field is expected to know how to use this data.

9 Deployment Strategies

Blooper App is containerized using Docker. These docker based containers are deployed on virtual machines (VMs) running on the cloud. The VMs are usually deployed on AWS, or Azure clouds.
Kubernetes too can be used to manage the deployment of these applications.

* * * * *

(Document prepared on 14 January, 2026 at 05:04:18 PM IST)