Implementation Document

for

Lostify

Version 1.0

Prepared by

Group #6 **Group Name:** **hAPPy Developers**

Aayush Kumar	230027	aayushk23@iitk.ac.in
Aman Raj	230116	amanraj23@iitk.ac.in
Anirudh Cheriyachanaseri Bijay	230140	anirudhcb23@iitk.ac.in
Ayush Patel	220269	ayushpatel22@iitk.ac.in
Krishna Kumayu	230576	kkrishna23@iitk.ac.in
Marada Teja Satvik	230636	maradateja23@iitk.ac.in
Satwik Raj Wadhwa	230937	satwikraj23@iitk.ac.in
Shaurya Johari	230959	shauryaj23@iitk.ac.in
Somaiya Narayan Aniruddh	231019	snarayana23@iitk.ac.in
Vinay Chavan	231155	vinay23@iitk.ac.in

Course: CS253

Mentor TA: Jeswaanth Gogula

Date: 28 March 2025

Contents

CONTENTS .... **II**

REVISIONS .... **III**

1 IMPLEMENTATION DETAILS **1**

1.1 FRONTEND 1
1.2 BACKEND 2
1.3 DATABASE 10
1.4 EXTERNAL SERVICES 11
1.5 BUILD SYSTEM 11

2 CODEBASE **12**

2.1 REPOSITORY 12
2.2 CODEBASE NAVIGATION 12

3 COMPLETENESS **19**

3.1 IMPLEMENTED FEATURES 19
3.2 FUTURE DEVELOPMENT PLANS 21

APPENDIX A – GROUP LOG **23**

Revisions

Version	Primary Author(s)	Description of Version	Date Completed
1.0	Aayush Kumar Aman Raj Anirudh Cheriyachanaseri Bijay Ayush Patel Krishna Kumayu Marada Teja Satvik Satwik Raj Wadhwa Shaurya Johari Somaiya Narayan Aniruddh Vinay Chavan	Preliminary draft	28/03/25

1 Implementation Details

1.1 Frontend

1.1.1 Programming Language

The frontend is written in Dart, an object-oriented, garbage-collected language inspired by C++, Java, and JavaScript among others. The choice for Dart is influenced by its close integration with the Flutter framework as well as ease of coding, its closeness to mainstream OOP languages, built-in null safety and memory safety, runtime type identification, and the availability of native debugging and performance tools (Dart DevTools).

1.1.2 Framework

The frontend uses the Flutter framework, which greatly simplifies app development with its fast performance and easy-to-understand hierarchy based on OOP principles. Key features include:

- **Widgets:** Flutter considers all UI elements widgets. This allows for uniformity amongst UI elements and a tree-style class hierarchy.
- **Stateful hot reload:** This revolutionary feature alleviates the need for compiling the source code from scratch to test any change in the code, a rather time-consuming process. Changes to UI elements do not change program state. For significant changes, a **stateful hot restart** can be performed.
- **Expressive UI:** Flutter provides a rich set of widgets and colour palettes conforming to the Material Design 3.0 specification with easy customization to suit the app theme, resulting in a beautiful yet familiar UI and an intuitive user experience.
- **Performance:** Flutter provides just-in-time (JIT) and ahead-of-time (AOT) compilation to enhance performance of applications.
- **Portability:** Dart code can, by itself, be compiled to native machine code, JavaScript, or WebAssembly. With the Flutter framework, Dart code can additionally be compiled to suit Android and iOS devices. Moreover, the user interface remains the same across various devices, screen sizes, and operating systems for the same class of devices.

1.1.3 Packages and Libraries

Flutter uses Dart's *pub.dev* package manager and software repository to extend functionality.

- **`flutter`:** Defines widgets corresponding to UI elements and colour palettes as described in the Material Design 3.0 specification.
- **`cloud_firestore`:** Handles cloud storage hosted by Firebase, exposing an API to read, write, and update documents stored.
- **`firebase_messaging`:** Manages real-time messaging between users supported by Firebase's Cloud Firestore service.

- `one_signal`: Provides real-time push notification service for mobile apps, integrating with the chat system.

- `http`: Sets up an HTTP client to connect and communicate with the backend using the request-response pattern.

- `provider`: Aids in state management, allowing efficient and scalable management of application state. It helps in maintaining separation between UI and business logic.

- `image_picker`: Used to select images from the device's gallery or capture photos using the camera, making it easy to handle media input in the application.

1.2 Backend

1.2.1 Programming Language

The backend is written in Python, a popular language for scripting and rapid development. With its rich set of libraries and built-in modules, duck typing paradigm, intuitive syntax that approaches natural language constructs, portability, and highly performant exception handling, Python has huge appeal for backend development.

1.2.2 Framework

The backend utilises the Flask framework written in Python, which eases backend development with several out-of-the-box features and libraries that abstract away most of the implementation detail while ensuring security and best practices. Some features that stand out are:

- **Simplicity and code readability:** Flask, being written in Python, inherits all its merits with regard to developer experience.

- **Security:** Flask defaults to secure, signed session cookies and comes packaged with the Werkzeug library, which implements cryptographically secure algorithms for secure authentication and storage of credentials such as salting and hashing of passwords.

- **Blueprints:** Flask provides code modularity through the concept of blueprints. Blueprints group routes related to each other (such as steps constituting an action or methods to access and modify a resource) together.

- **Inbuilt support for URL routing:** Flask handles URL routing through function decorators, mitigating boilerplate to call route-related functions and to parse parameters embedded in the URL path.

1.2.3 Packages and Libraries

Libraries included with Flask are combined with the Python Standard Library and external libraries handled through pip, Python's package manager.

- `flask`: Defines widgets corresponding to UI elements and colour palettes as described in the Material Design 3.0 specification.

- **`werkzeug`:** Implements the WSGI interface in full conformance with the Python specification. The `werkzeug.security` module provides security-related features in adherence to cryptographic standards.

- **`python_dotenv`:** Manages environment variables defining paths to configuration files and local storage and API keys for cookie signing and external API authentication.

- **`azure.communication.email`:** Provides a Python interface to Microsoft Azure Email Communication Services to mail OTPs to users.

- **`sqlite3`:** Exposes the SQLite3 API through a Python interface to handle SQLite3 databases. Conforms to the Python Database API Specification v2.0.

- **`secrets`:** Includes the `SystemRandom` class, which generates random numbers using the highest-quality sources provided by the operating system. This is used for OTP generation.

1.2.4 API Endpoints

The backend exposes a RESTful API that the frontend communicates with using the HTTP protocol. This design enables a complete rewrite of the frontend without affecting the backend; similarly, backend logic can be altered without affecting the frontend. The implementation of each is opaque to the other.

The API exposes the following endpoints accessible through the specified URLs and HTTP methods:

1.2.4.1 Authentication

URL `/auth/signup/get_otp`

Method POST

Purpose Initiate a registration request and send an OTP.

Request Data MIME type: `application/json`

```
{
    username,
    password,
    profile
}
```

Response Status Success:
- **201:** OTP sent

Failure:
- **400:** Type mismatch for JSON field(s)
- **400:** Username must be alphanumeric
- **400:** Field is required
- **405:** Method not allowed

- **409:** User already exists

URL `/auth/signup/verify_otp`

Method POST

Purpose Complete a registration request by verifying the OTP supplied by the user.

Request Data MIME type: `application/json`

```
{
    username,
    otp
}
```

Response Status Success:
- **201:** User created

Failure:
- **400:** Type mismatch for JSON field(s)
- **400:** Username is required
- **401:** Incorrect OTP
- **404:** Username not found
- **404:** OTP timed out
- **405:** Method not allowed

URL `/auth/login`

Method POST

Purpose Authenticate a user and set session cookies upon successful verification of credentials.

Request Data MIME type: `application/json`

```
{
    username,
    password
}
```

Response Status Success:
- **200:** User authenticated

Failure:
- **400:** Field is required
- **401:** Incorrect password
- **404:** Username not found
- **405:** Method not allowed

- **429:** Login attempt limit reached

URL `/auth/logout`

Method GET, HEAD

Purpose Clear session cookies.

Request Data None.

Response Status Success:

- **205:** Successful logout; reset content

URL `/auth/change_password`

Method POST

Purpose Change the account password.

Request Data MIME type: `application/json`

```
{
    old_password,
    new_password
}
```

Response Status Success:

- **204:** Password changed

Failure:

- **400:** Old/New password is required
- **401:** User not logged in
- **401:** Incorrect password
- **404:** Username not found
- **405:** Method not allowed

URL `/auth/reset_password`

Method POST

Purpose Reset the account password if forgotten.

Request Data MIME type: `application/json`

```
{
    username
}
```

Response Status

Success:
- **204:** New password sent to email

Failure:
- **404:** User not found
- **405:** Method not allowed

1.2.4.2 Post Handling

URL `/items/post`

Method POST

Purpose Create a new post.

Request Data MIME type: `application/json`

```
{
    type,
    title,
    description,
    location1,
    location2,
    image
}
```

Response Status

Success:
- **201:** Post created successfully

Failure:
- **400:** Field is required
- **400:** Type must be either 0 or 1
- **401:** User not logged in
- **405:** Method not allowed

URL `/items/<int:id>`

Method GET

Purpose Retrieve details of the post with post id `id`.

Request Data None.

Response Status

Success:
- **200:** Post data in response body as JSON

Failure:
- **401:** User not logged in
- **404:** Post not found

URL `/items/<int:id>`

Method PUT

Purpose Update the post with post id `id`.

Request Data MIME type: `application/json`

```
{
    title,
    description,
    location1,
    location2,
    image
}
```

Response Status

Success:
- **204:** Post updated successfully

Failure:
- **400:** No fields to update
- **400:** Field is required
- **401:** User not logged in
- **403:** User is not creator of post

URL `/items/<int:id>`

Method DELETE

Purpose Delete the post with post id `id`.

Request Data None.

Response Status

Success:
- **204:** Post deleted successfully

Failure:
- **401:** User not logged in
- **403:** User is not authorised to delete post
- **404:** Post not found

URL `/items/all`

Method GET

Purpose Retrieve details of all posts at once.

Request Data None.

Response Status Success:

- **200:** Post data in response body as JSON

Failure:

- **401:** User not logged in

URL `/items/<int:id>/claim`

Method GET

Purpose Claim the post with post id `id`.

Request Data None.

Response Status Success:

- **200:** Status of claimed post in response body as JSON

Failure:

- **400:** ID of second party is required
- **401:** User not logged in
- **404:** Post not found
- **405:** Method not allowed
- **409:** Post already claimed

URL `/items/<int:id>/report`

Method GET

Purpose Retrieve count of reports of post with post id `id`.

Request Data None.

Response Status Success:

- **200:** Report count in response body as JSON

Failure:

- **401:** User not logged in

- **403:** User is not authorised to view report count
- **404:** Post not found

URL	`/items/<int:id>/report`
Method	PUT
Purpose	Report post with post id `id`.
Request Data	None.
Response Status	Success: • **204:** Reported

URL	`/items/<int:id>/report`
Method	DELETE
Purpose	Undo report of post with post id `id`.
Request Data	None.
Response Status	Success: • **204:** Deleted report

1.2.4.3 Profile management

URL `/users/<int:id>/profile`

Method PUT

Purpose Update the profile of the user whose id is `id`.

Request Data MIME type: `application/json`

```
{
    name,
    phone,
    email,
    address,
    designation,
    roll,
    image
}
```

Response Status	Success: • **204:** Profile updated successfully Failure: • **400:** No fields being updated • **400:** Name is required • **401:** User not logged in • **403:** Profile does not belong to user

URL	`/users/<int:id>/profile`
Method	GET
Purpose	Retrieve the profile details of the user whose id is `id`.
Request Data	None.
Response Status	Success: • **200:** Profile details in response body as JSON Failure: • **401:** User not logged in • **404:** User not found

1.3 Database

The backend relies on a relational database model handled using SQLite3, a C library implementing a lightweight and compact yet performant SQL database engine featuring ACID transactions and a cross-platform file format. SQLite3 is popular for small- and medium-sized applications.

The advantages of SQLite3 over traditional client-server RDBMS systems are:

- **Ease of installation and use:** The Python interface to SQLite3 is part of the Python Standard Library; consequently, no additional installation requirements arise in the use of SQLite3 with Flask.

- **Low memory footprint:** SQLite3 databases are stored as a single file and its handling can be done entirely through the SQLite3 API.

- **Platform independence:** SQLite3 database files are platform-independent and can be moved to other platforms without change.

- **High performance for small- and medium-sized applications:** While traditional, enterprise RDBMS systems are built to handle large workloads with heavy traffic, huge amounts of data, and many concurrent requests, SQLite3 is designed to be performant in the context of small- or medium-sized applications without enormous traffic or large storage requirements.

1.4 External Services

1.4.1 Firebase

Lostify's chat feature has been implemented using Firebase, leveraging its real-time database capabilities to enable seamless messaging between users.

Cloud Firestore (or Realtime Database): It is used to store the chat messages for each chat. Its real-time synchronization capability is employed to ensure that messages are instantly delivered and displayed to users as they are sent.

1.4.2 Cloudinary

Cloudinary is integrated into the application to handle image uploads in the chat messges. The choice of this service is driven by its robust cloud services, significantly lower costs, and ease of integration.

Cloudinary comes with great ease of integration, permitting authorized clients to upload and store data through simple API requests.

1.4.3 OneSignal

To provide timely updates and enhance user engagement, push notification functionality was implemented using OneSignal.

Integration with Chat System: When a message is added, the recipient's Player ID (previously stored in Database, linked to the user profile) is extracted and a notification is dispatched to the recipient via OneSignal through an API request.

1.4.4 Microsoft Azure

Microsoft Azure hosts the domain, email address, and email server from which OTPs are dispatched to users' registered email addresses and exposes a Python API so that the Flask backend can supply the email content and recipient's email address and request the SMTP server to send an email.

1.5 Build System

The Flutter frontend uses the Gradle build system that comes packaged with Android Studio to automate the compile-build-debug process for Android targets.

2 Codebase

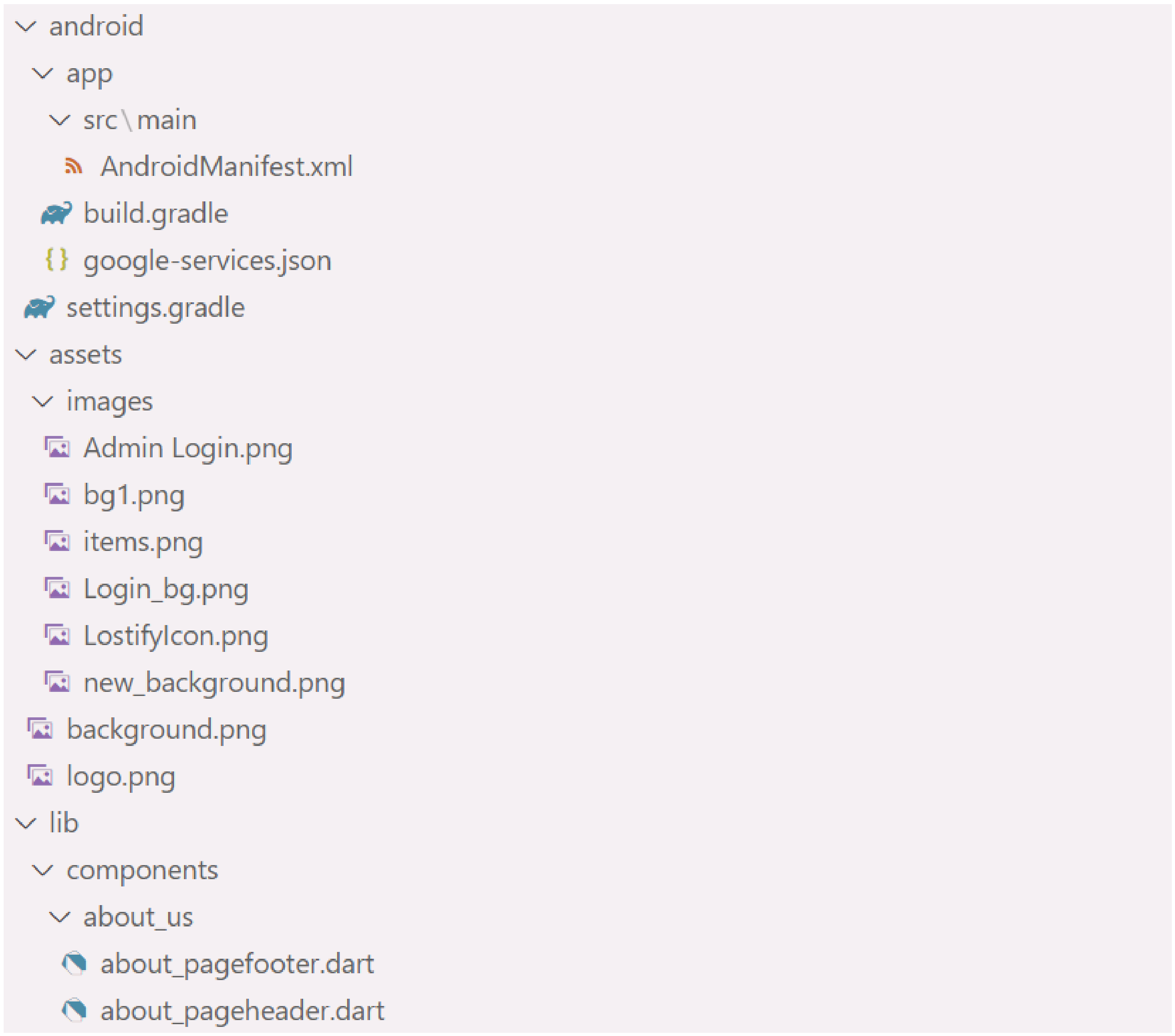
2.1 Repository

The source code for both the frontend and the backend is hosted on GitHub. The repository can be found at the link https://github.com/CS253-Group-6/Lostify.

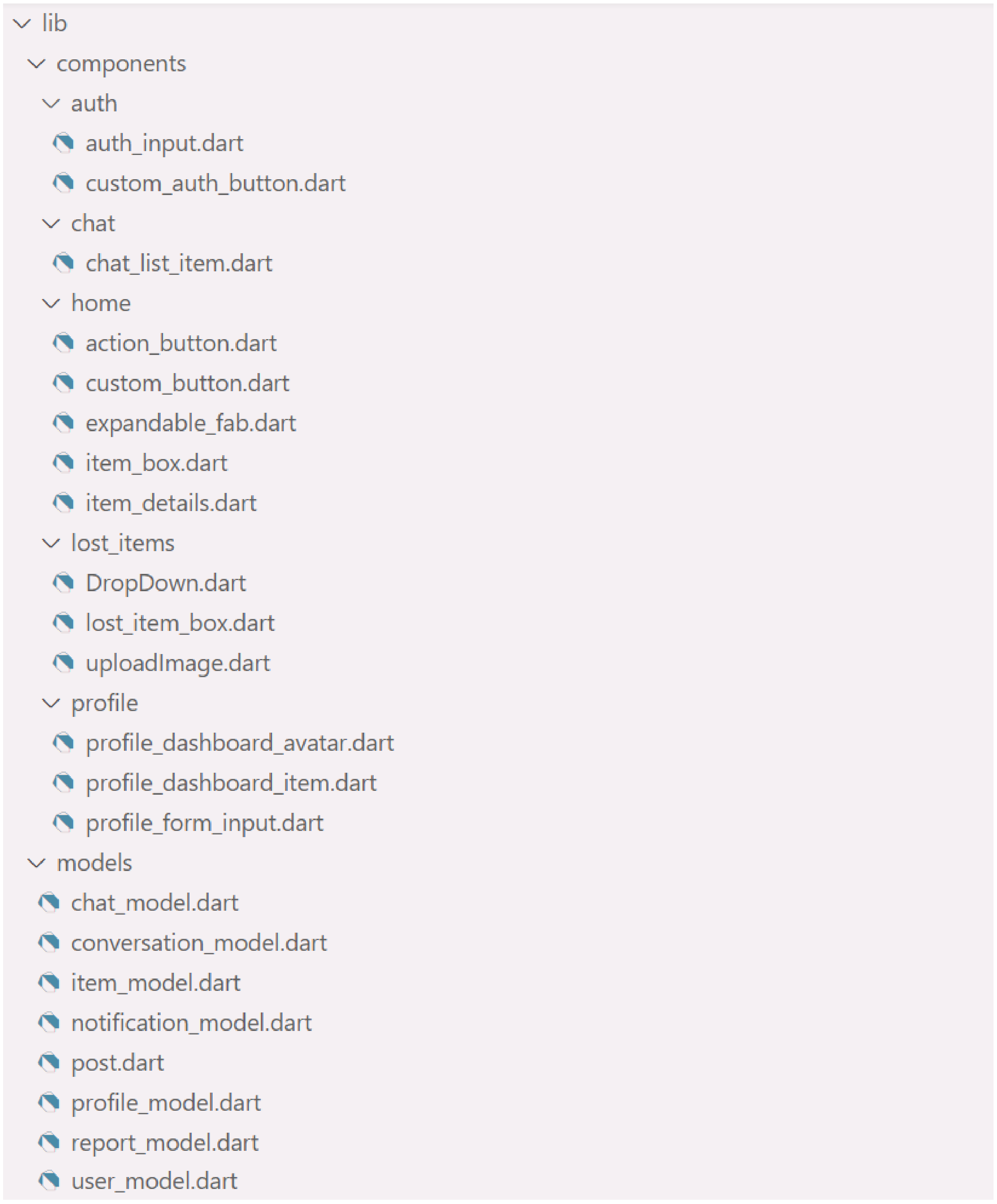
2.2 Codebase Navigation

2.2.1 Frontend Directory

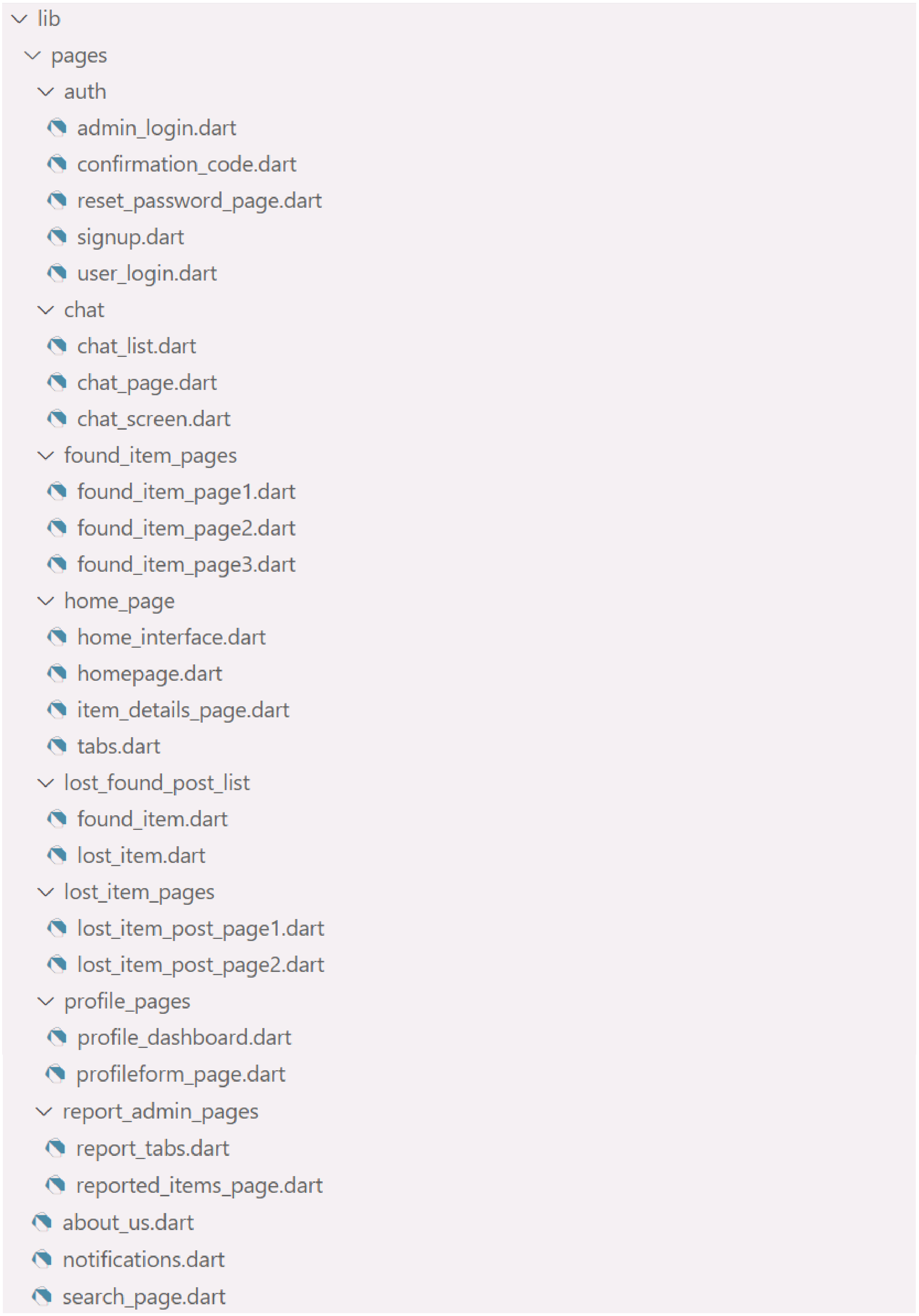
The overall directory structure of the frontend is automatically generated by Android Studio's project management tools for Flutter. The below directory structure mentions only those directories and files that have been created or modified for the purposes of the project.



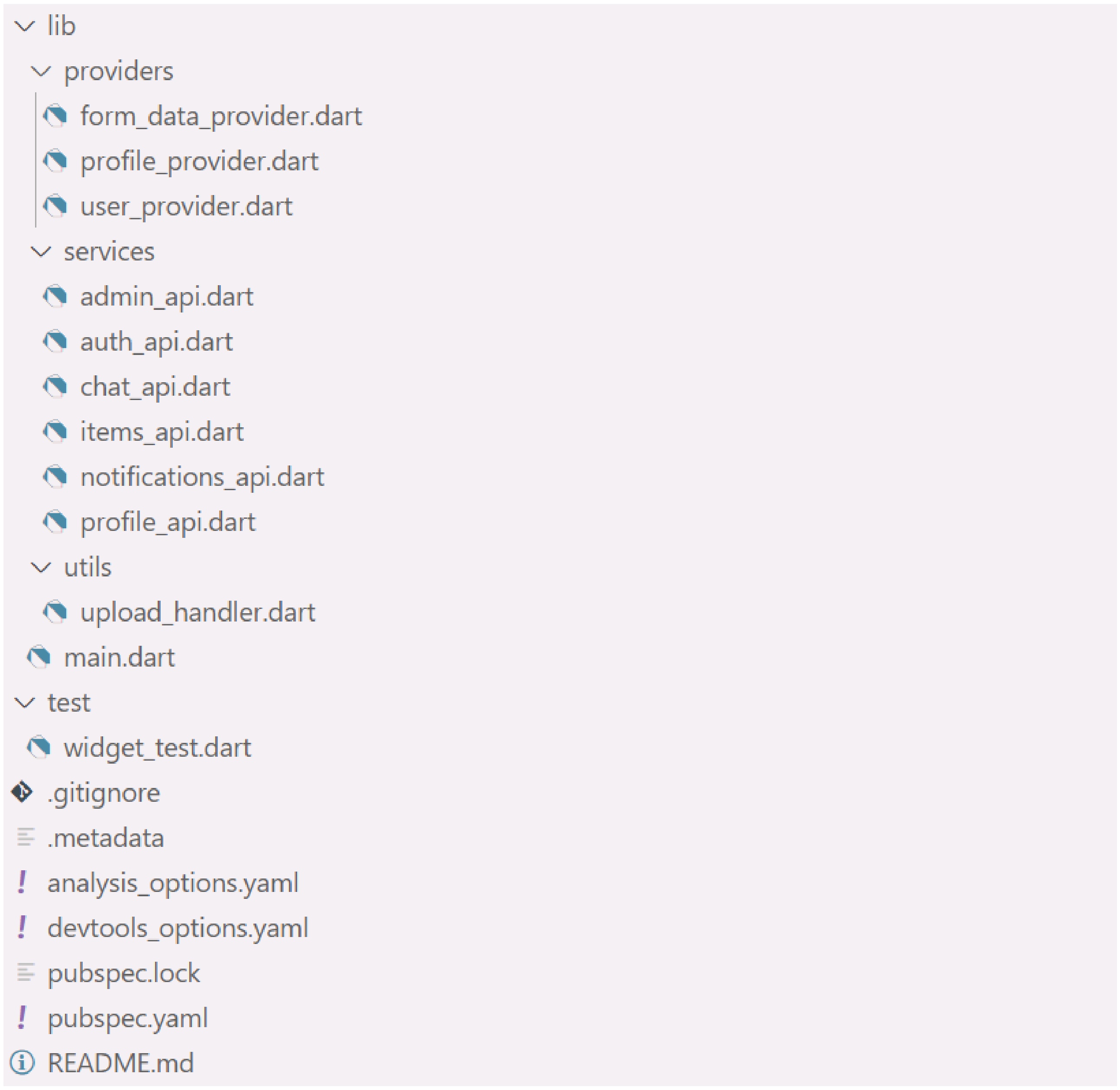
```
android
  app
    src\main
      AndroidManifest.xml
    build.gradle
    google-services.json
  settings.gradle
assets
  images
    Admin Login.png
    bg1.png
    items.png
    Login_bg.png
    LostifyIcon.png
    new_background.png
  background.png
  logo.png
lib
  components
    about_us
      about_pagefooter.dart
      about_pageheader.dart
```



```
lib
  components
    auth
      auth_input.dart
      custom_auth_button.dart
    chat
      chat_list_item.dart
    home
      action_button.dart
      custom_button.dart
      expandable_fab.dart
      item_box.dart
      item_details.dart
    lost_items
      DropDown.dart
      lost_item_box.dart
      uploadImage.dart
    profile
      profile_dashboard_avatar.dart
      profile_dashboard_item.dart
      profile_form_input.dart
  models
    chat_model.dart
    conversation_model.dart
    item_model.dart
    notification_model.dart
    post.dart
    profile_model.dart
    report_model.dart
    user_model.dart
```



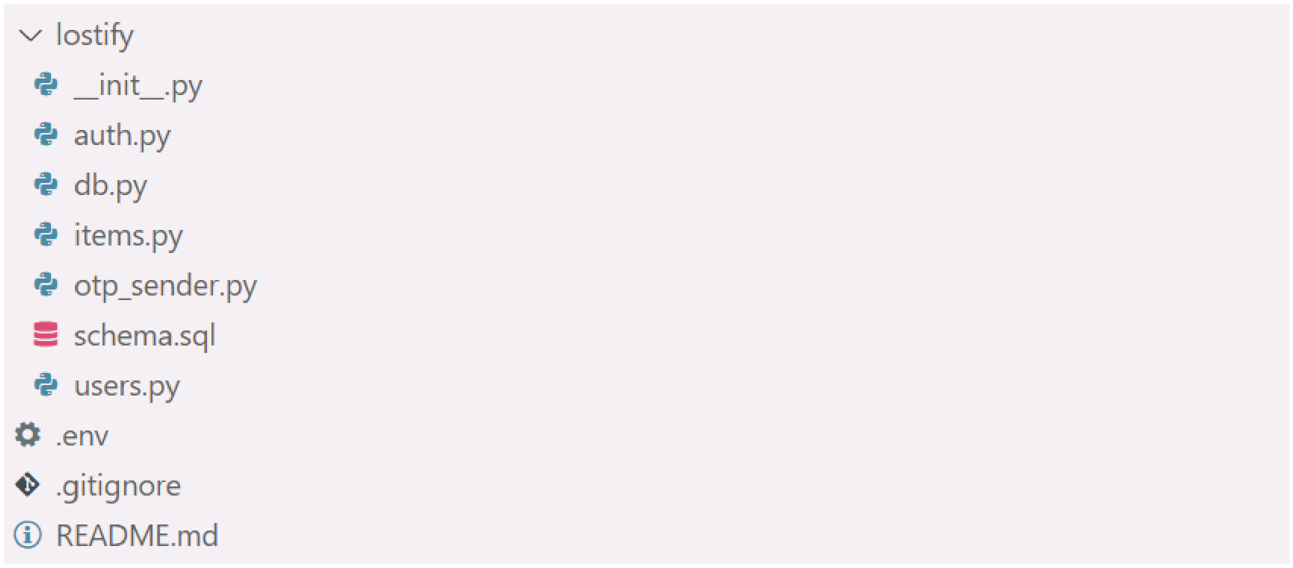
```
lib
  pages
    auth
      admin_login.dart
      confirmation_code.dart
      reset_password_page.dart
      signup.dart
      user_login.dart
    chat
      chat_list.dart
      chat_page.dart
      chat_screen.dart
    found_item_pages
      found_item_page1.dart
      found_item_page2.dart
      found_item_page3.dart
    home_page
      home_interface.dart
      homepage.dart
      item_details_page.dart
      tabs.dart
    lost_found_post_list
      found_item.dart
      lost_item.dart
    lost_item_pages
      lost_item_post_page1.dart
      lost_item_post_page2.dart
    profile_pages
      profile_dashboard.dart
      profileform_page.dart
    report_admin_pages
      report_tabs.dart
      reported_items_page.dart
    about_us.dart
    notifications.dart
    search_page.dart
```



```
lib
  providers
    form_data_provider.dart
    profile_provider.dart
    user_provider.dart
  services
    admin_api.dart
    auth_api.dart
    chat_api.dart
    items_api.dart
    notifications_api.dart
    profile_api.dart
  utils
    upload_handler.dart
  main.dart
test
  widget_test.dart
.gitignore
.metadata
analysis_options.yaml
devtools_options.yaml
pubspec.lock
pubspec.yaml
README.md
```

2.2.2 Backend Directory

The backend contains a `.env` file containing environment variables such as API keys in the root directory. Flask code is placed under the folder titled `lostify`; however, the code must be run from the root directory itself.



```
lostify
  __init__.py
  auth.py
  db.py
  items.py
  otp_sender.py
  schema.sql
  users.py
.env
.gitignore
README.md
```

2.2.3 Databases

2.2.3.1 `awaitOTP`

Field Name	Type	Constraint	Function

`username`	`Text`	`Primary Key`
		`Not Null`
			Email address of user

`password`	`Text`	`Not Null`	Password
`otp`	`Integer`	`Not Null`	Otp
`created`	`Integer`	`Not Null`	Time of OTP generation
`profile`	`Text`	`Not Null`	Profile Details

2.2.3.2 `users`

Field Name	Type	Constraint	Function

`id`	`Integer`	`Primary Key`
		`Autoincrement`
			Email address of user

`username`	`Text`	`Unique`
		`Not Null`
			Name

`password`	`Text`	`Not Null`	Password
`role`	`Integer`	`Not Null`	0 for students 1 for admin

`counter`	`Integer`	`Not Null`
		`Default 0`
			Number of failed login attempts

`lastAttempt`	`Integer`	`Not Null`	Time of last unsuccessful attempt

2.2.3.3 `profiles`

Field Name	**Type**	**Constraint**	**Function**

`userid`	`Integer`	`Primary Key`
		`Not Null`
		`Foreign Key: users(id)`	

`name`	`Text`	`Not Null`	Name of the user
`phone`	`Text`		Phone number
`email`	`Text`		Email address
`address`	`Text`		Residence address
`designation`	`Text`		Designation

`roll`	`Integer`	`Unique`
		`Not Null`
			Roll number

`image`	`BLOB`		Profile image

2.2.3.4 `posts`

Field Name	**Type**	**Constraint**	**Function**

`id`	`Integer`	`Primary Key`
		`Autoincrement`
			Email address of user

`type`	`Integer`	`Not Null`	0 for lost 1 for found
`creator`	`Integer`	`Not Null`	User id of the post creator
`title`	`Text`	`Not Null`	Post title
`description`	`Text`		Post description
`location1`	`Text`	`Not Null`	Coarse location of find/loss
`location2`	`Text`		Fine location of find/loss
`image`	`BLOB`	`Image`	Image of article
`date`	`Integer`	`Not Null`	Date of post creation
`closedBy`	`Integer`	`Not Null`	User id of claimant
`closedDate`	`Integer`	`Not Null`	Date of closing post

`reportCount`	`Integer`	`Not Null`
		`Default 0`
			Count of reports

2.2.3.5 `reports`

Field Name	**Type**	**Constraint**	**Function**
`postid`	`Integer`	`Not Null`	Post id of reported post
`userid`	`Integer`	`Not Null`	User id of reporter

2.2.3.6 `confirmations`

Field Name	**Type**	**Constraint**	**Function**
`postid`	`Integer`	`Not Null`	Post id
`Initid`	`Integer`	`Not Null`	Id of initiator

`otherid`	`Integer`	`Not Null`	Id of user from whom confirmation is pending

2.2.4 Setting up the Development Environment

2.2.4.1 Frontend

1. Clone the repository from GitHub.
2. Navigate to the frontend directory and open it in Android Studio.
3. Run `flutter pub get` to get the updated dependencies.

2.2.4.2 Backend

1. Clone the repository from GitHub.
2. Navigate to the backend directory and create a virtual environment.
3. Activate the environment in the terminal.
4. Install the dependencies specified in the `requirements.txt` file using `pip install -r requirements.txt`.
5. Create a `.env` file with the database path and API keys.
6. Start the development server with `flask –app lostify run`.

3 Completeness

3.1 Implemented Features

3.1.1 Home Interface

When the app starts, the home screen appears with three options:

- **User Login** – Takes users to the login page.
- **Admin Login** – Directs administrators to the admin login page.
- **Not a member? Register now –** Guides new users through the registration process.

3.1.2 Login

Login requires an email address and password for authentication.

The login page includes a 'Forgot Password' button, which redirects users to the password reset flow.

Upon successful login, users are redirected to the homepage**.**

3.1.3 Password Reset

Users initiate the password reset process by entering their registered email address, which redirects to the confirmation code page.

- An OTP (One-Time Password) is sent to the given email for verification.
- The confirmation code page provides an option to resend the OTP if necessary.
- Upon successful verification, users can set a new password, after which they are redirected to the home page.

3.1.4 Registration

- Clicking 'Not a member? Register now' redirects to the Sign-up Page, where users provide:
    - Email Address
    - Password
- On the next page, users proceed to the Create Profile Page, where they enter:
    - Display Name
    - Phone Number
    - Campus Address
    - Designation
    - Roll Number
    - Profile Image
- Upon submitting profile details, users are redirected to the page for OTP verification.
- After successful verification, users are redirected to the home page.

3.1.5 Homepage and Dashboard

The home page contains the following elements:

- A floating action button that expands into two floating action buttons when tapped:
    - Post a Lost Item.
    - Post a Found Item.
- Tabs:
    - **Lost** – Displays only lost item chats.
    - **Found** – Displays only found item chats.
- Hamburger icon: Opens a dashboard to navigate to other parts of the app.
- Search icon: Opens the search page.

3.1.6 Posting Lost and Found Items

Opens a form where the user provides:

- A title for the post
- Description of the article
- Image of the article
- Location where the item was lost/Location where the item found can be collected
- Date and time of the loss/find

3.1.7 Own Posts

The user can view his/her own posts and manage them from two sections:

- **Your Lost Items** – Shows all chats related to the user's lost items.
- **Your Found Items** – Displays all chats related to the user's found items.

This page is reached from the dashboard.

3.1.8 Messages

The messages page is a centralised view of the user’s chats with other users regarding posts. This page can be reached from the dashboard.

3.1.9 Search

The user can search for a post through the search facility provided.

- The user navigates to the search page from the homepage by tapping the search icon on the app bar.
- The user uses the following fields to refine their search:
    - Location
    - Date range

3.1.10 View Reported Items

Users have the facility to flag posts. Admins can view a list of reported posts sorted in descending order of count of users who reported them. They may also delete posts from this page.

This page can be navigated to from the dashboard; the relevant option appears only for admins.

3.1.11 Edit Profile

The user can edit his/her profile at any time.

- From the dashboard, the user opens the Edit Profile page.
- The user may edit any of the following details:
  - Display Name
  - Phone Number
  - Campus Address
  - Designation
  - Roll Number
  - Profile Image
- The user may then submit the form, whereupon his/her profile is updated with the new details.

3.1.12 Logout

The user can log out from the dashboard; he/she is then redirected to the home interface.

3.2 Future Development Plans

3.2.1 Google Maps API integration

- **Live Item Location Pinning** – Users can mark the last known location of lost items.
- **Real-Time Directions to Found Items** – Guide users to lost items reported nearby.

3.2.2 Community engagement and partnerships

- **Extend to other institutions** – The application can be customised for use by other educational institutions, offices and similar establishments.
- **Gamification and Rewards** – Introduce points or incentives for users who report found items.

3.2.3 AI-powered image and text recognition

- **Image Matching** – AI-based image recognition to match found items with lost reports automatically.
- **OCR (Optical Character Recognition)** – Extract details from images of receipts, tags, or engravings on objects.
- **Categorisation** – Autogenerate descriptions for items from their images.

3.2.4 Priority Recognition

- **Urgency Detection** – Items like passports or wallets can be prioritised and displayed higher up in the homepage and search results.

3.2.5 Porting to iOS

- **Future Expansion to iOS** – Lostify is Android-only now, but Flutter allows easy transition to iOS.

- **Uniform UI/UX Across Platforms** – Ensure consistency in design and features.
- **Apple Maps Alternative for iOS** – Option to switch between Google Maps and Apple Maps.

Appendix A – Group Log

Date & Time	Venue	Minutes
07/03/25 11:00 pm	4th floor, Rajeev Motwani Building	Discussed implementation iork, decided the work and team meet timelines, and distributed work among teammates. Decided the tentative dates for completion of frontend and backend components, as well as the start date for integration of both ends.
09/03/25 3:00 pm	Online	Gathered resources for learning Flask and Flutter and set up our machines for the development process. Redesigned the logo for the app. Basic layout of the app refined for development.
13/03/25 9:00 pm	Online	Revision of components and distribution of individual pages of the application.
17/03/25 9:00 am	1st floor, Rajeev Motwani Building	Met to compile our work and analyse our progress and plan future development processes. Discussed more about the Flask backend routes and decide the app's routes.
19/03/25 10:00 pm	1st floor, Rajeev Motwani Building	Further progress on frontend/learning routes and other backend-related knowledge. Learned more about the APIs necessary to facilitate backend. Revised plans and timelines for backend implementation.
22/03/25 9:00 pm	1st floor, Rajeev Motwani Building	Finished the frontend. Began working on the backend and integration. Started working on the implementation document. Distributed the backend work amongst teammates.
26/03/25 10:00 am	1st floor, Rajeev Motwani Building	Completed backend and frontend. Started working on the integration process. Finalized the design for the databases needed for the code.
27/03/25 6:00 pm	1st floor, Rajeev Motwani Building	Further discussion on integrating frontend/backend. Work on user manual started. Work on testing document started.
28/03/25 2:00 pm	1st floor, Rajeev Motwani Building	Finalized the Implementation Document. Successfully established important routes of the app. Made progress in integration and uploaded the necessary files on GitHub.