C3F1

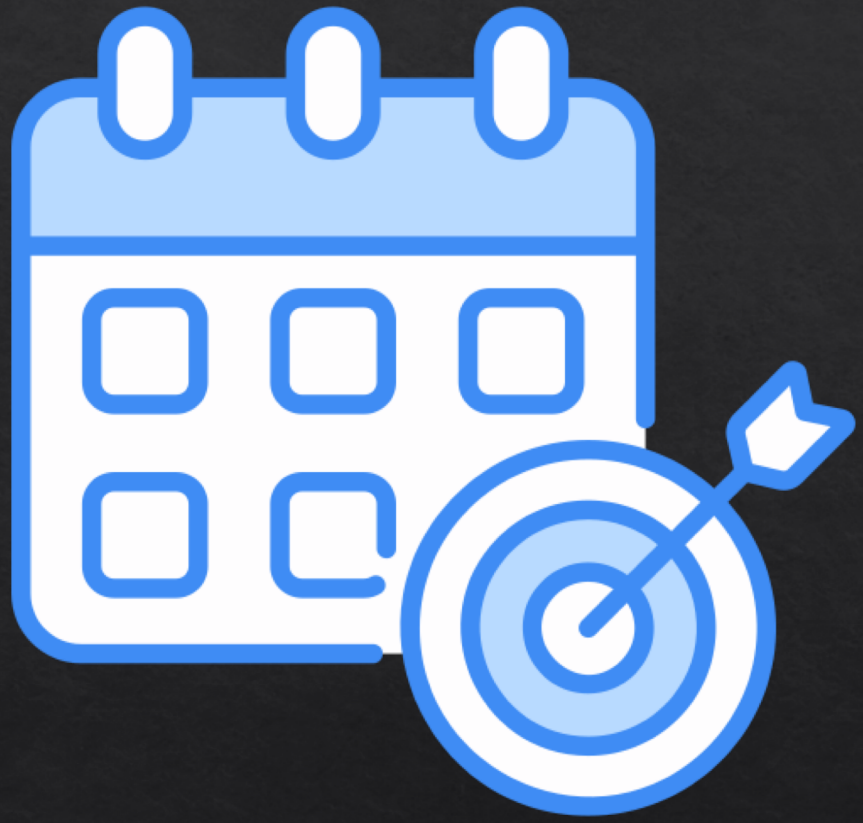
Project proposed by:

Calò Domenico, Capuano Marzia, Crispino Luca, Ficarella Fabrizio

A.A. 2024/2025

MAIN OBJECTIVE

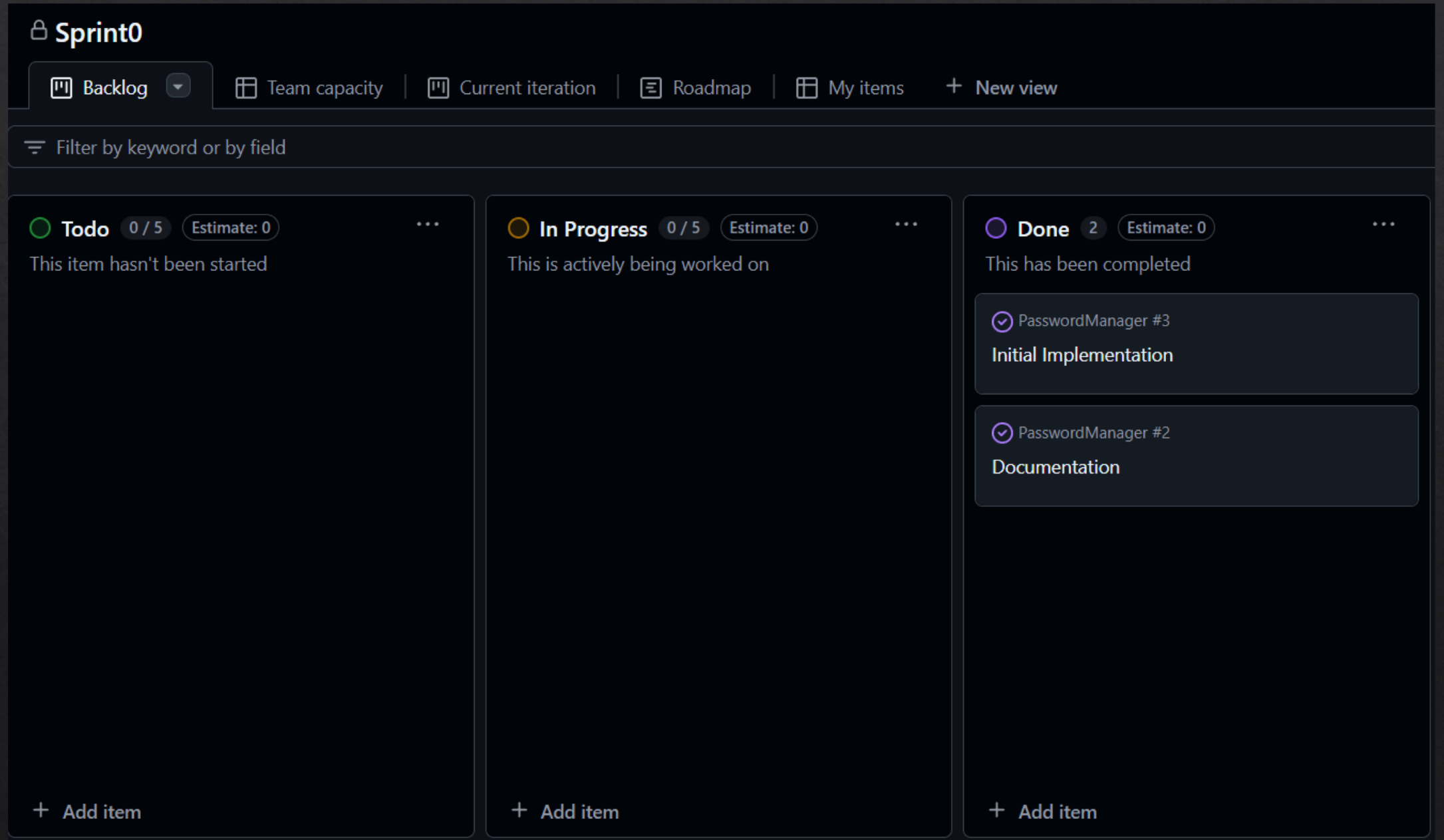
The project consists of developing a secure and user-friendly Password Manager designed to help individuals and organizations store, manage, and generate strong passwords and other secrets.



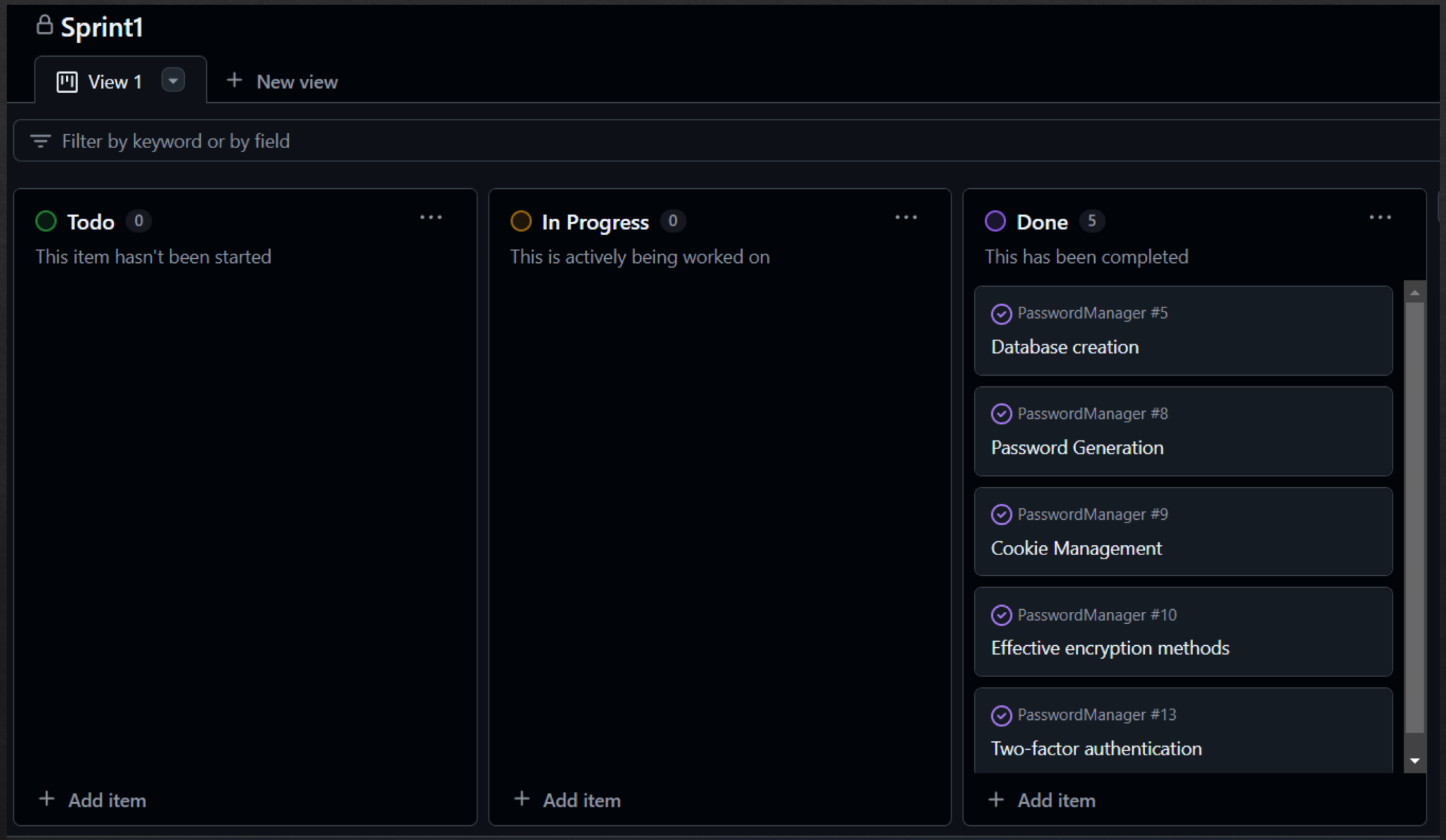
CHALLENGES

- Effective encryption methods
- Password Generation
- Accessible everywhere
- Stores all types of sensitive data
- Two-factor authentication

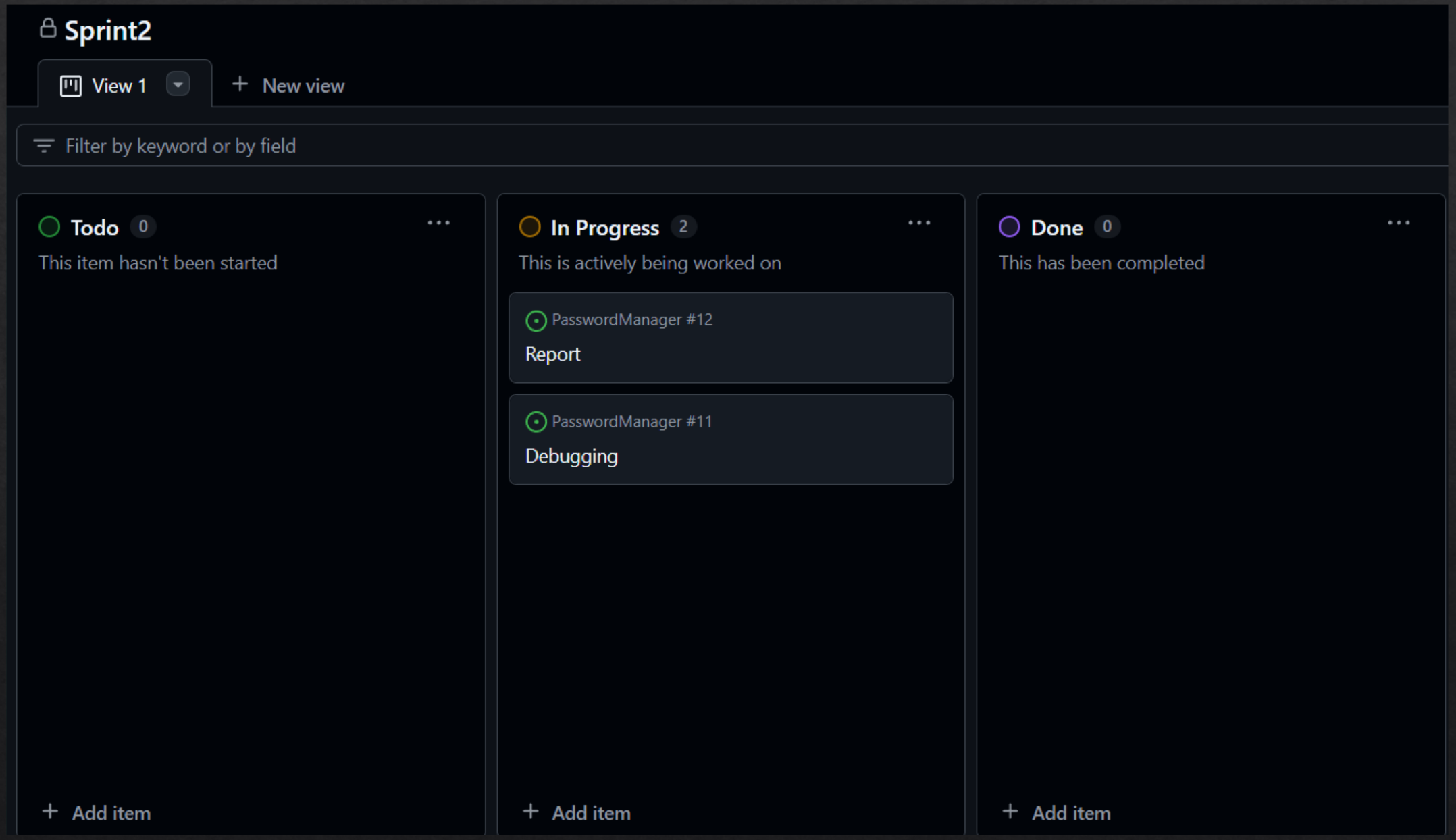
Progress of the project



Progress of the project



Progress of the project



Database Creation (SQLAlchemy)

```python
class User(db.Model, UserMixin):
    id = db.Column(db.Integer, primary_key=True)
    username = db.Column(db.String(80), unique=True)
    password = db.Column(db.String(150))
    email = db.Column(db.String(120), unique=True)
    site_data = db.relationship( *args: 'SiteData', backref='user')
    created_at = db.Column(db.DateTime, nullable=False)
    is_two_factor_authentication_enabled = db.Column(
        db.Boolean, nullable=False, default=False)
    otp_token = db.Column(db.String, unique=True)
```

Table: user Page: 0 Jump << < 1-1 > >> Refresh

id	userna...	passwo...	email	created...	is_two_...	otp_tok...
1	l.crispino	scrypt:32...	lucacrispi...	2024-11-...	1	7OY4YS...
2	staticUse...	scrypt:32...	staticEm...	2024-12-...	0	Z3M6POI...
3	Marzia	scrypt:32...	marziaca...	2024-12-...	0	WBVSUK...
4	Marzia1	scrypt:32...	marziaca...	2024-12-...	0	ROMPBS...

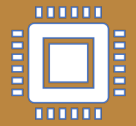
Database Creation (SQLAlchemy)

```python
class SiteData(db.Model):  9 usages
    id = db.Column(db.Integer, primary_key=True)
    site = db.Column(db.String(150))
    username = db.Column(db.String(150))
    email = db.Column(db.String(150))
    password = db.Column(db.LargeBinary)
    user_id = db.Column(db.Integer, db.ForeignKey('user.id'), nullable=False)
```

Table: site_data Page: 0 Jump << < 1-1 > >> Refresh

id	site	userna...	email	passwo...	user_id
1	dfgh	dfgh	dfg	BLOB	1

PROPOSED SOLUTION



For an optimal solution, it was decided to use a cryptography based on the Python's **Fernet** library that allows different types of effective solutions such as the standard AES 128.

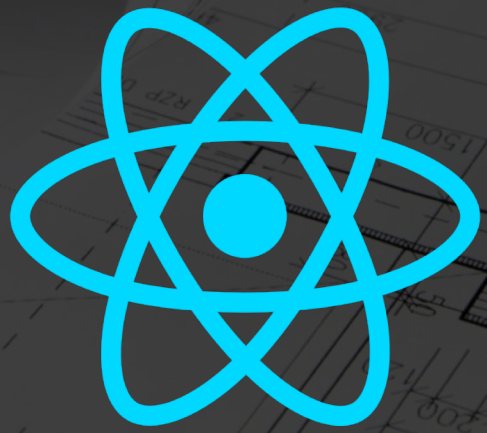
The idea is to generate at first use of the app a **Master Key** that allows the user to encrypt and decrypt data containing sensitive information.

SOFTWARE ARCHITECTURE

To create our password manager, we'll use principally:

- Flask for the backend
- React for the frontend

Our app we'll use Google Authenticator to generate valid TOTPs.



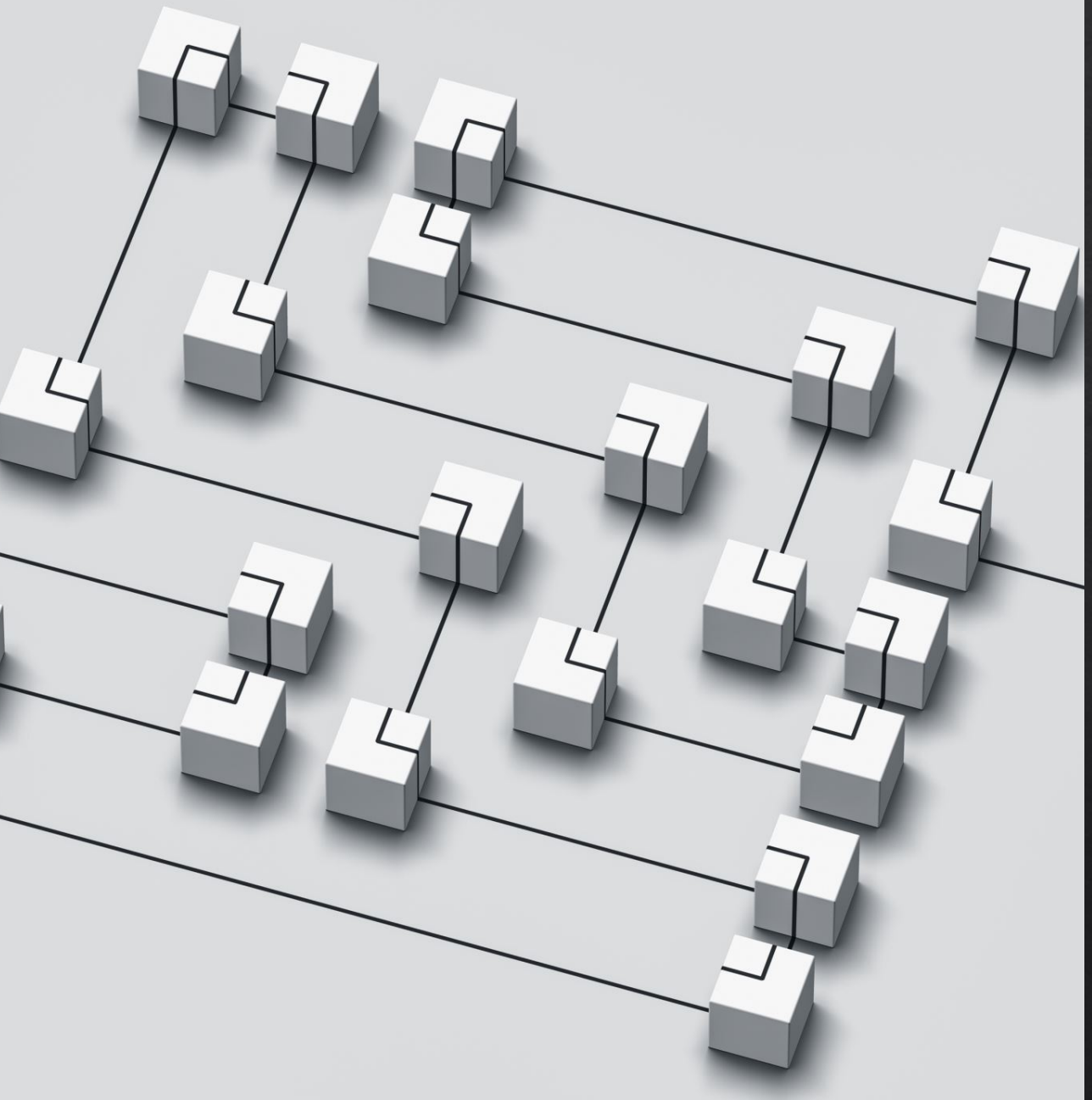
REQUIREMENTS

FUNCTIONAL

- Random passwords generation
- TOTP verification
- Secure storage
- Cross-platform usage

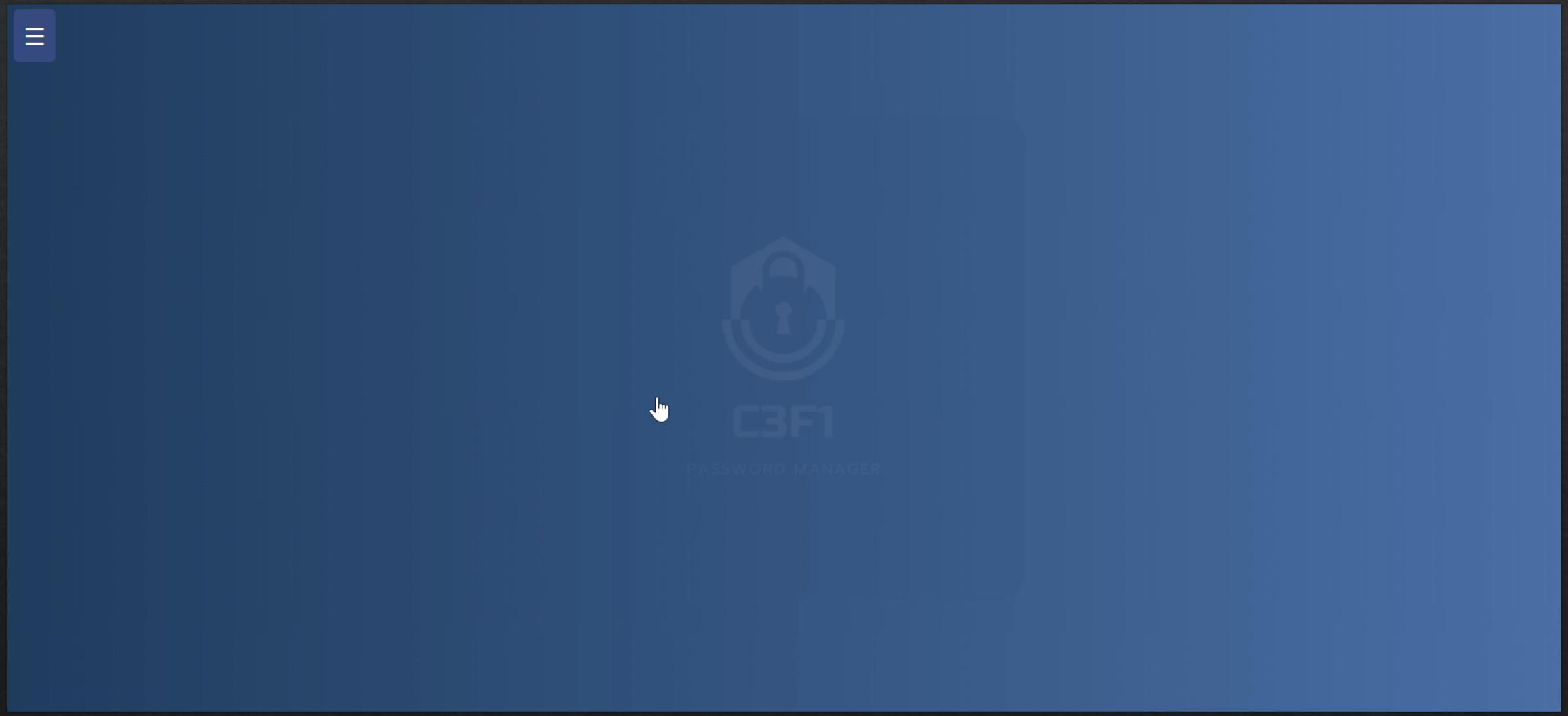
NON-FUNCTIONAL

- Security
- Reliability
- Portability
- Usability



DESIGN SCHEME

- At the beginning, you will be presented with a page where you can choose between logging in or registering.
- If you register, you must be logged in to access the app. In this case, you will be redirected to the profile setup page for Google Authenticator
- The OTP verification page can be accessed either after login (for logins after the first one) or after setup (for the first login)
- If the OTP code provided is correct, you will be able to access the home page where you can perform all possible operations


C3F1
PASSWORD MANAGER

A.A. 2024/2025