```
extern crate bitwarden;
extern crate reqwest;
extern crate rpassword;

#[macro_use]
extern crate clap;

fn main() {
    //let password = "p4ssw0rd";
    //let email = "nobody@example.com";
    //let salt = email.to_lowercase();
    //let master_key = bitwarden::make_key(password, &salt);
    //println!( "Hello, masterKey:{}", String::from_utf8_lossy(&master_key.unwrap()));
    //let protected_key = bitwarden::make_encrypted_key(master_key.unwrap());
    //println!("encrypted_key: {}", protected_key);
    //let master_password_hash = bitwarden::hashed_password(password, email);
    //println!("master_password_hash:{}", master_password_hash);
    //let register_body = bitwarden::signup(&email, &master_password_hash, &protected_key);
    //println!("signup:{}", register_body);
    //let client = reqwest::Client::new();
    // let res = client
    //    .post("http://127.0.0.1:8000/api/accounts/register")
    //    .body(register_body)
    //    .send();
    //println!("register post:{:?}", res);
    let matches = clap::App::new("Bitwarden CLI")
        .version(crate_version!())
        .author(crate_authors!())
        .arg(
            clap::Arg::with_name("url")
                .help("URL of bitwarden server")
                .takes_value(true)
                .default_value("http://127.0.0.1:8000")
                .short("u"),
        )
        .subcommand(
            clap::SubCommand::with_name("register")
                .about("register new account with bitwarden server.")
                .arg(
                    clap::Arg::with_name("email")
                        .help("email address to login as.")
                        .required(true),
                ),
        )
        .subcommand(
            clap::SubCommand::with_name("login")
                .about("login to bitwarden")
                .arg(
                    clap::Arg::with_name("email")
                        .help("email address to login as.")
                        .required(true),
                ),
        )
        .subcommand(
            clap::SubCommand::with_name("sync")
            .about("sync against remote bitwarden server")
        )
        .get_matches();

    let url = reqwest::Url::parse(matches.value_of("url").unwrap()).unwrap();

    match matches.subcommand() {
        ("register", Some(register_matches)) => {
            // Now we have a reference to register's matches
            let email = register_matches.value_of("email").unwrap().to_lowercase();
            println!("Register: {}", email);
            let password = rpassword::prompt_password_stdout("Password: ").unwrap();
            let result = "";
            //let result = bitwarden::register(&url, &email, &password);
            let result = bitwarden::register(&url, &email, &password);
            println!("result:{}", result);
        }
        ("login", Some(login_matches)) => {
            // Now we have a reference to login's matches
            let email = login_matches.value_of("email").unwrap().to_lowercase();
            println!("Login:{}", email);
            let password = rpassword::prompt_password_stdout("Password: ").unwrap();
            let result = bitwarden::login(&url, &email, &password);
            println!("result:{}", result);
        }
        ("sync", Some(sync_matches)) => {
            // Now we have a reference to login's matches
            // let token= "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJlbWFpbCI6Im5vYm9keUBleGFtcGxlLmNvbSIsImV4cCI
6MTUyMjg4NDIzMywiaXNzIjoiTkEiLCJuYW1lIjoiIiwibmJmIjoxNTIyODgwNjMzLCJwcmVtaXVtIjpmYWxzZSwic3ViIjoiTkEifQ.46xvH6-
FQKNuFWOVXLeeut3bvHE2QMSUg45aH557XRU";
            //let result = bitwarden::sync(&url, &token);
```

```
85              //println!("login result:{}", result);
86
87          }
88          ("add", Some(add_matches)) => {
89              // Now we have a reference to add's matches
90              println!(
91                  "Adding {}",
92                  add_matches
93                      .values_of("stuff")
94                      .unwrap()
95                      .collect::<Vec<_>>()
96                      .join(", ")
97              );
98          }
99          ("", None) => println!("No subcommand was used"), // If no subcommand was used it'll match the tuple ("", None)
100         _ => unreachable!(), // If all subcommands are defined above, anything else is unreachable!()
101     }
102 }
```