TASKMATE

BY

KUSH PALKESH DUDHIA
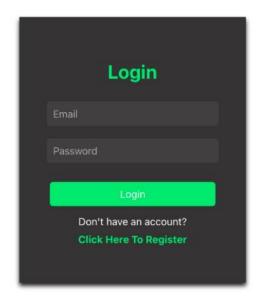
KRITHIKA REDDY SANGIREDDY

ANSHU REDDY ASHANNA
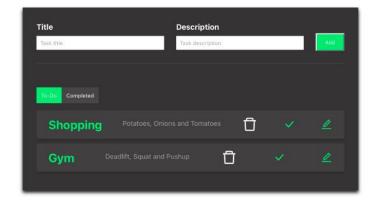
NAIYANI PALADUGU

DEMONSTRATION

INTRODUCTION

- The project focuses on creating a robust and user-friendly task management application.

- Built using Scala for high performance and the Play Framework for scalability.

- Addresses the need for efficient task management for individuals and teams.

OVERVIEW

- **Objective**: Develop a web-based to-do list application.

- **Features**:

- Create, edit, delete, and view tasks.

- Mark tasks as complete or pending.

- Provide secure user authentication and authorization.

- **End Goal**: Deliver a functional, scalable, and user-friendly tool.

TOOLS AND TECHNOLOGIES

- **Programming Language**: Scala for functional programming and performance.

- **Framework**: Play Framework for building robust web applications.

- **Database**: PostgreSQL for data persistence.

- **Development Tools**: VSCode or IntelliJ IDEA for coding and debugging.

CORE FEATURES

- **User Authentication**: Ensure data security with login and user-specific task management.

- **CRUD Operations**: Seamlessly create, read, update, and delete tasks.

- **Data Persistence**: Save tasks persistently using PostgreSQL.

- **Responsive UI**: Intuitive and user-friendly interface for ease of use.

- **REST API**: Enable integration with other applications or tools.

ARCHITECTURAL OVERVIEW

User.scala ✕

app > models > User.scala

```scala
1   package models
2
3   import anorm._
4   import play.api.libs.json._
5
6   case class User (
7       username: String,
8       password: String
9   )
10
11  object User{
12    implicit def toParameters: ToParameterList[User] = Macro.toParameters[User]
13
14  implicit val implicitWrites = new Writes[User] {
15      def writes(user: User): JsValue = {
16          Json.obj(
17            "username" -> user.username,
18            "password" -> user.password,
19          )
20        }
21      }
22  }
```

RootController.scala ✕

app > controllers > RootController.scala

```scala
1   package controllers
2
3   import javax.inject._
4   import play.api._
5   import play.api.mvc._
6   import play.api.data._
7   import play.api.data.Forms._
8   import play.api.i18n.I18nSupport
9   import models.Task
10  import repositories.TaskRepository
11
12  /**
13   * This controller creates an `Action` to han
14   * application's home page.
15   */
16  @Singleton
17  class RootController @Inject()(taskService: T
18
19
20    def index = Action { implicit request: Requ
21      Redirect(routes.TaskController.index)
22    }
23
24  }
25
```

```
∨ TASKMATE-SCALA
  > .bloop
  > .g8
  > .metals
  > .vscode
  ∨ app
    ∨ controllers
      > api
      ≡ RootController.scala
      ≡ TaskController.scala
      ≡ UserController.scala
    ∨ models
      ≡ DB.scala
      ≡ Main.scala
      ≡ Task.scala
      ≡ User.scala
    > repositories
    > views
  > conf
  > gradle
  > project
  > public
  > spec
```

LIMITATIONS

- **Compatibility Issues**: Aligning Java and Scala versions for seamless integration.

- **Debugging**: Identifying and resolving bugs in Scala, which may be less intuitive.

- **Integration**: Combining Play Framework with third-party tools or APIs.

BUSINESS USE CASES

- **Productivity**: Improve task prioritization and efficiency for individuals and teams.

- **Time Management**: Help users track deadlines and stay organized.

- **Process Improvement**: Streamline workflows by identifying inefficiencies.

- **Scalability**: Adaptable for individual users or large teams in businesses.

FUTURE ENHANCEMENTS

- **Task Prioritization**: Allow users to categorize and prioritize tasks.
- **Reminders & Notifications**: Ensure timely task completion with alerts.
- **Collaboration**: Support multiple users working on shared task lists.
- **Mobile App**: Extend functionality to Android and iOS platforms.
- **Tool Integration**: Enable syncing with calendars or productivity tools.