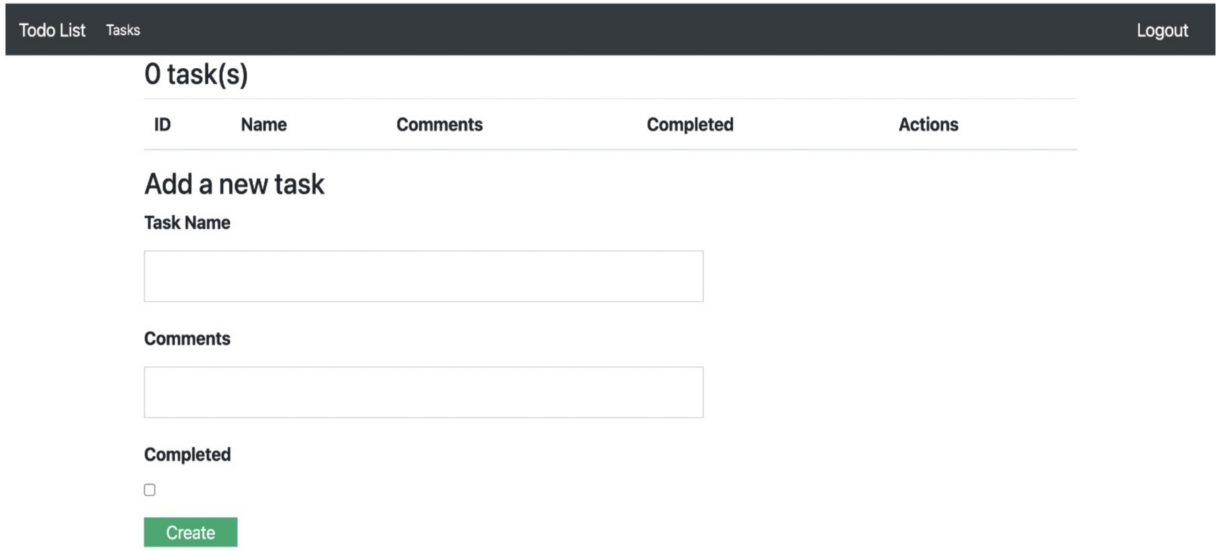
TO-DO LIST USING SCALA

By:-
Bhavya Sree
Rithik Reddy
S SAITEJA
Prima Sunil
Praveen Chowdary

DEMONSTRATION

DEMO


Todo List Tasks Logout

0 task(s)

ID	Name	Comments	Completed	Actions

Add a new task

Task Name

Comments

Completed

Create

PROOF OF CONCEPT

- The Scala to-do list web application is a project that aims to create a simple and intuitive task management tool for users. The application is built using the Scala programming language and utilizes the Play Framework to provide a robust and scalable web application architecture.
- The project involves creating a web-based interface where users can create, edit, and delete tasks in their to-do lists. The application will also provide users with the ability to mark tasks as complete and view completed tasks. Additionally, the application will have user authentication and authorization features to ensure that users can securely manage their tasks.
- The end goal of the project is to create a functional, user-friendly, and scalable to-do list application that can be easily deployed and used by individuals and teams.

TOOLS USED

- Scala
- Play framework
- VSCode
- PostgreSQL

FEATURES AND FUNCTIONALITIES

- ➢ User Authentication and Authorization
- ➢ CRUD Operations
- ➢ Data Persistence
- ➢ User Interface
- ➢ REST Api

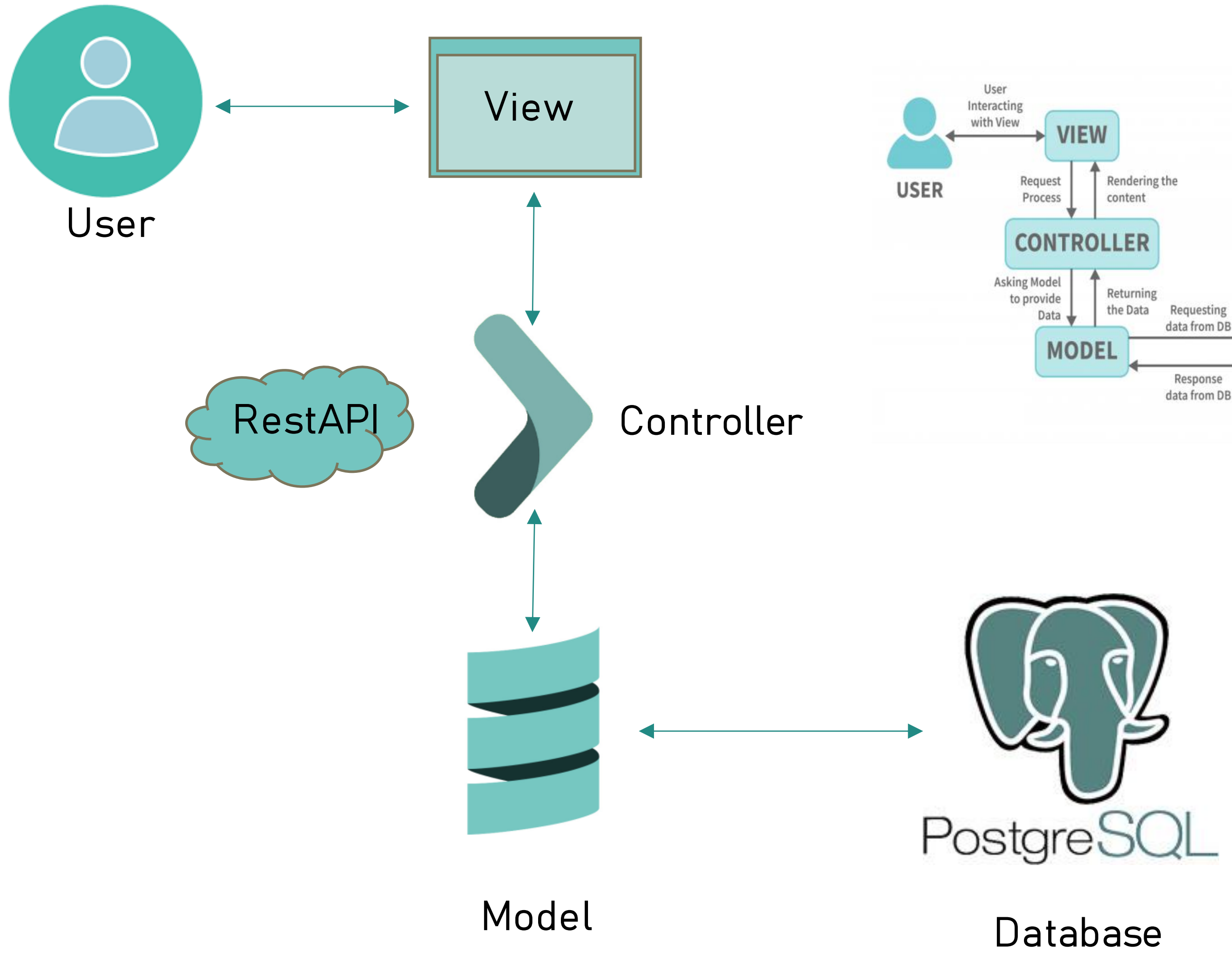
BUSINESS PROSPECTS

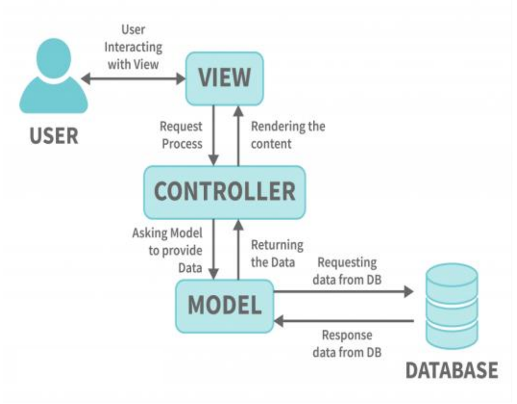
The business function of this to-do list project is to enhance the efficiency and productivity of employees or teams in a business by helping them prioritize their work and manage their time effectively. It can also identify areas for improvement in task management, leading to better results and streamlined processes.

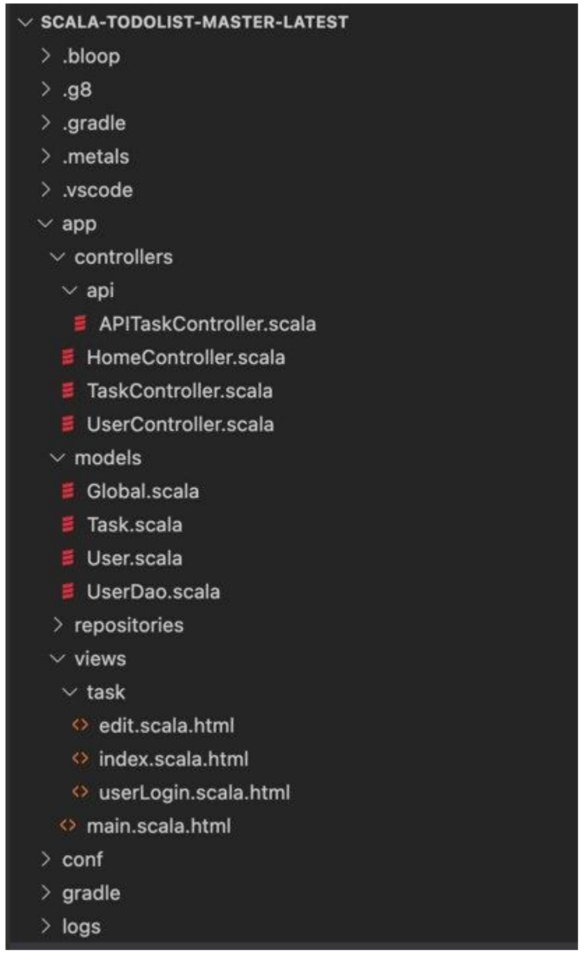
ROADBLOCKS IN PROJECT IMPLEMENTATION

- Java and scala version compatibility Issue
- Debugging can be more challenging in Scala
- Integration challenges with existing framework.

ARCHITECTURAL PATTERN

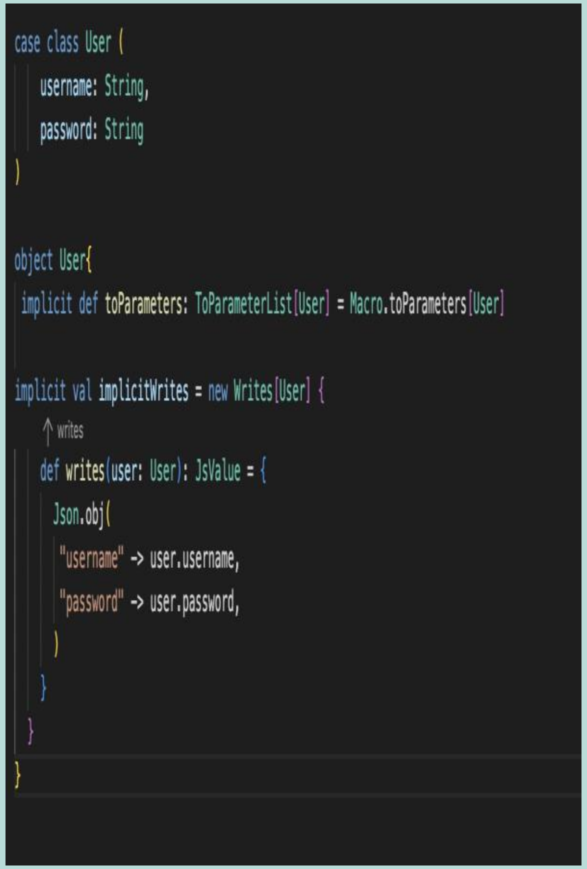






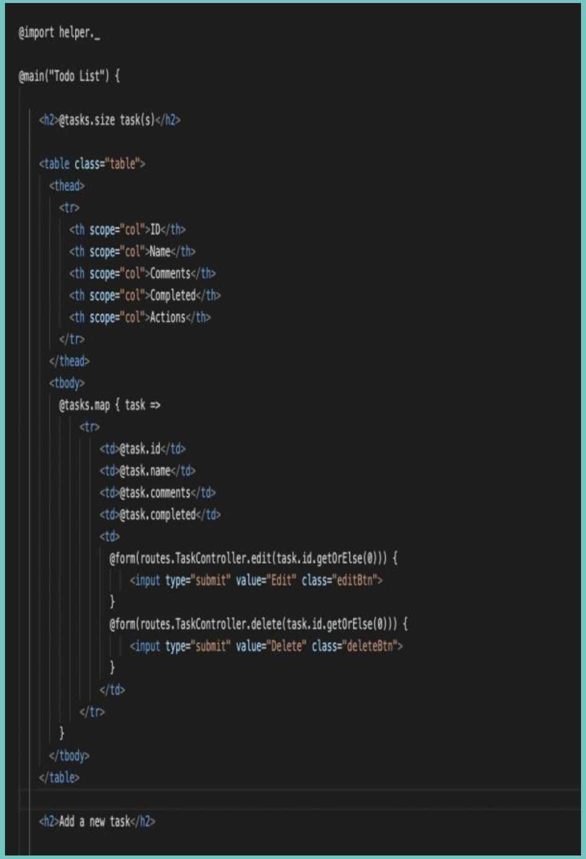
```
SCALA-TODOLIST-MASTER-LATEST
  > .bloop
  > .g8
  > .gradle
  > .metals
  > .vscode
  v app
    v controllers
      v api
          APITaskController.scala
        HomeController.scala
        TaskController.scala
        UserController.scala
    v models
        Global.scala
        Task.scala
        User.scala
        UserDao.scala
    > repositories
    v views
      v task
          edit.scala.html
          index.scala.html
          userLogin.scala.html
        main.scala.html
  > conf
  > gradle
  > logs
```

MODEL



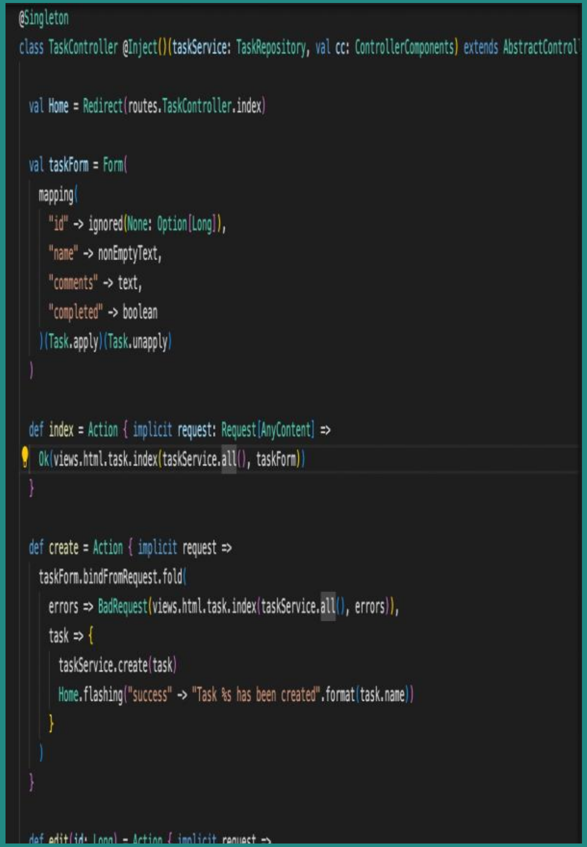
```scala
case class User (
  username: String,
  password: String
)

object User{
 implicit def toParameters: ToParameterList[User] = Macro.toParameters[User]

implicit val implicitWrites = new Writes[User] {
    ↑ writes
  def writes(user: User): JsValue = {
    Json.obj(
     "username" -> user.username,
     "password" -> user.password,
    )
  }
 }
}
```

VIEW



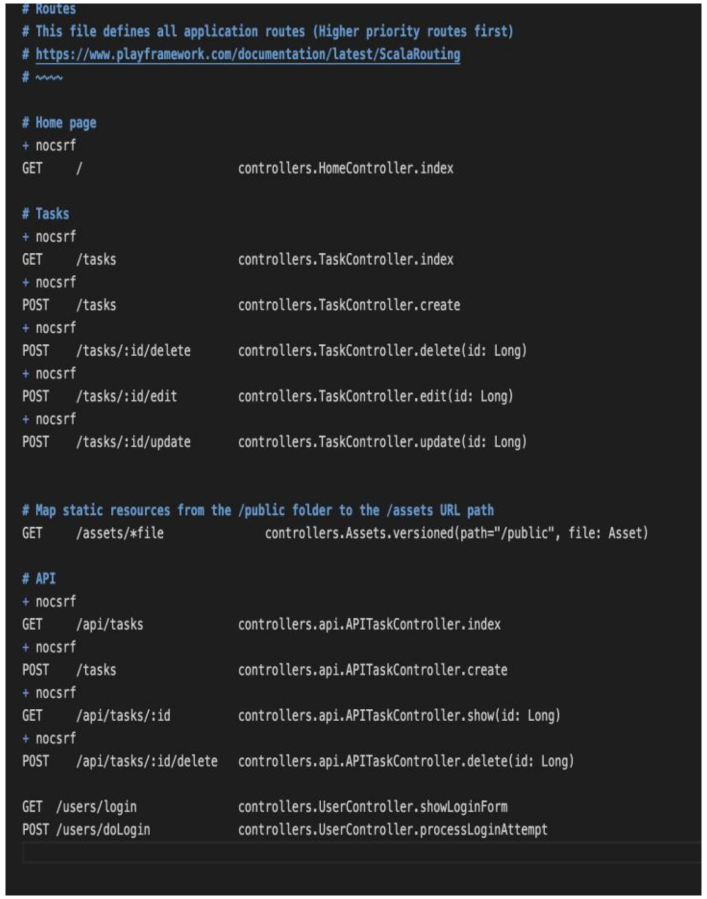
```html
@import helper._

@main("Todo List") {

   <h2>@tasks.size task(s)</h2>

   <table class="table">
     <thead>
       <tr>
         <th scope="col">ID</th>
         <th scope="col">Name</th>
         <th scope="col">Comments</th>
         <th scope="col">Completed</th>
         <th scope="col">Actions</th>
       </tr>
     </thead>
     <tbody>
       @tasks.map { task =>
           <tr>
               <td>@task.id</td>
               <td>@task.name</td>
               <td>@task.comments</td>
               <td>@task.completed</td>
               <td>
                 @form(routes.TaskController.edit(task.id.getOrElse(0))) {
                     <input type="submit" value="Edit" class="editBtn">
                 }
                 @form(routes.TaskController.delete(task.id.getOrElse(0))) {
                     <input type="submit" value="Delete" class="deleteBtn">
                 }
               </td>
           </tr>
       }
     </tbody>
   </table>

   <h2>Add a new task</h2>
```

CONTROLLER

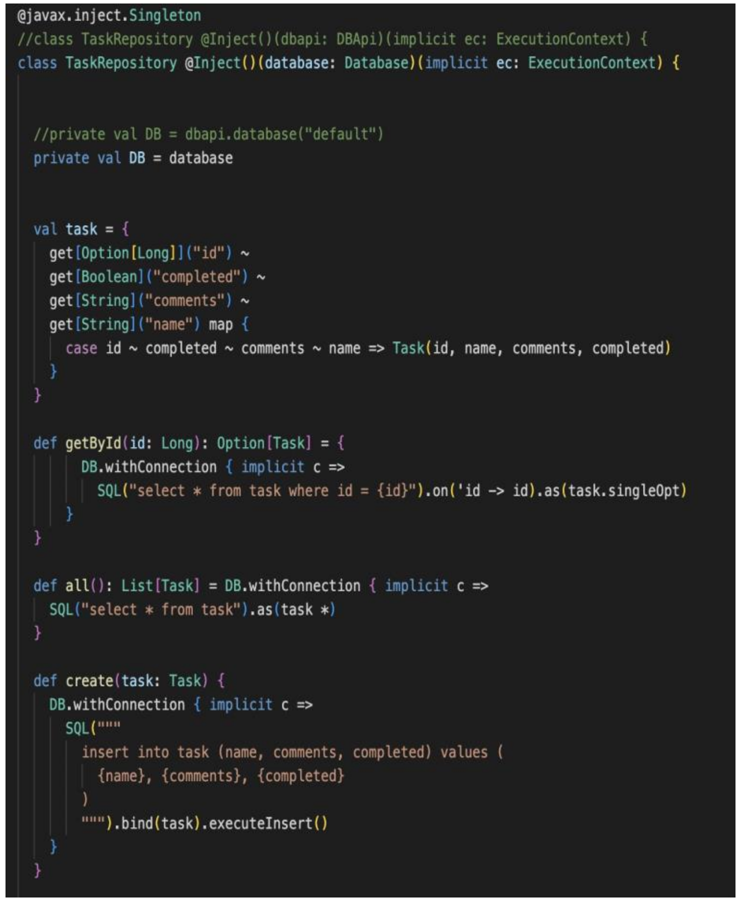


```scala
@Singleton
class TaskController @Inject()(taskService: TaskRepository, val cc: ControllerComponents) extends AbstractControl

  val Home = Redirect(routes.TaskController.index)

  val taskForm = Form(
    mapping(
      "id" -> ignored(None: Option[Long]),
      "name" -> nonEmptyText,
      "comments" -> text,
      "completed" -> boolean
    )(Task.apply)(Task.unapply)
  )

  def index = Action { implicit request: Request[AnyContent] =>
    Ok(views.html.task.index(taskService.all(), taskForm))
  }

  def create = Action { implicit request =>
    taskForm.bindFromRequest.fold(
      errors => BadRequest(views.html.task.index(taskService.all(), errors)),
      task => {
        taskService.create(task)
        Home.flashing("success" -> "Task %s has been created".format(task.name))
      }
    )
  }

  def edit(id: Long) = Action { implicit request =>
```

PROJECT IMPLEMENTATION



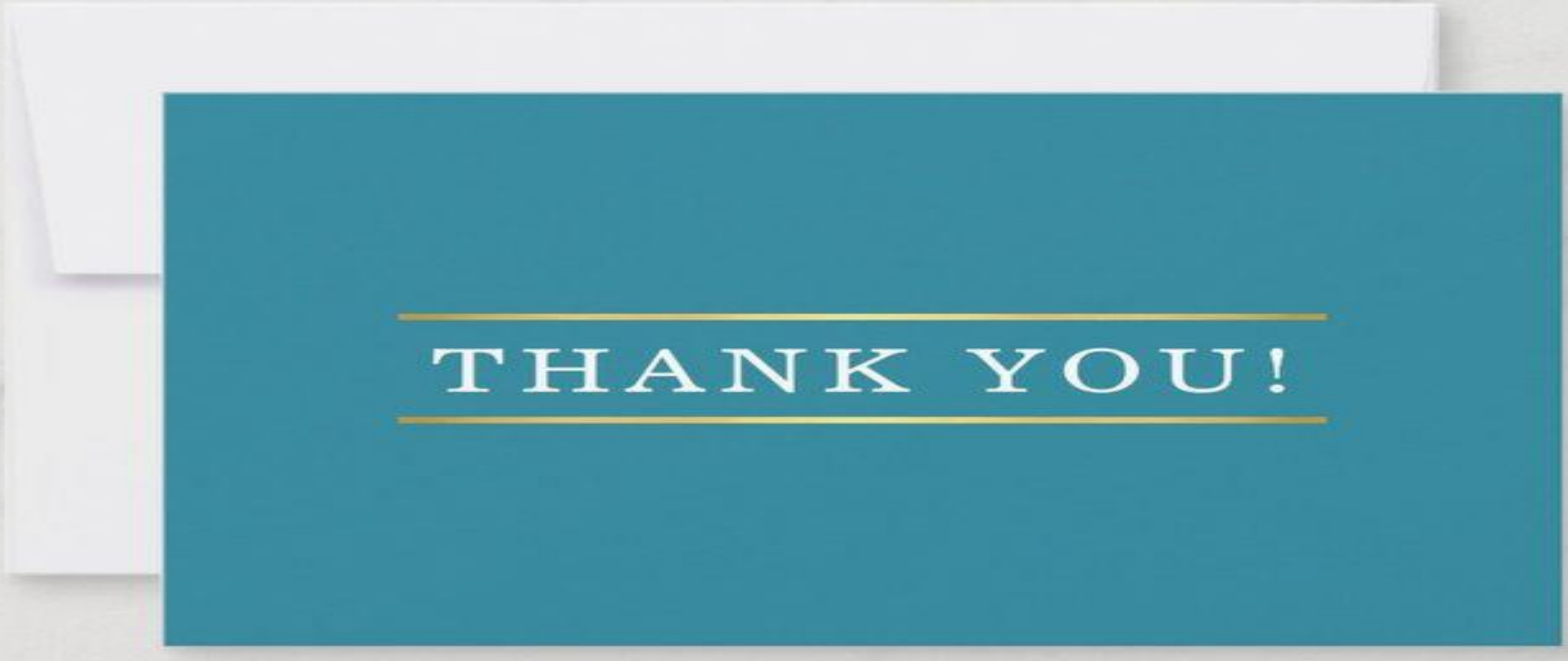
```
# Routes
# This file defines all application routes (Higher priority routes first)
# https://www.playframework.com/documentation/latest/ScalaRouting
# ~~~~

# Home page
+ nocsrf
GET     /                       controllers.HomeController.index

# Tasks
+ nocsrf
GET     /tasks                  controllers.TaskController.index
+ nocsrf
POST    /tasks                  controllers.TaskController.create
+ nocsrf
POST    /tasks/:id/delete       controllers.TaskController.delete(id: Long)
+ nocsrf
POST    /tasks/:id/edit         controllers.TaskController.edit(id: Long)
+ nocsrf
POST    /tasks/:id/update       controllers.TaskController.update(id: Long)


# Map static resources from the /public folder to the /assets URL path
GET     /assets/*file               controllers.Assets.versioned(path="/public", file: Asset)

# API
+ nocsrf
GET     /api/tasks              controllers.api.APITaskController.index
+ nocsrf
POST    /tasks                  controllers.api.APITaskController.create
+ nocsrf
GET     /api/tasks/:id          controllers.api.APITaskController.show(id: Long)
+ nocsrf
POST    /api/tasks/:id/delete   controllers.api.APITaskController.delete(id: Long)

GET  /users/login               controllers.UserController.showLoginForm
POST /users/doLogin             controllers.UserController.processLoginAttempt
```



```scala
@javax.inject.Singleton
//class TaskRepository @Inject()(dbapi: DBApi)(implicit ec: ExecutionContext) {
class TaskRepository @Inject()(database: Database)(implicit ec: ExecutionContext) {


  //private val DB = dbapi.database("default")
  private val DB = database


  val task = {
    get[Option[Long]]("id") ~
    get[Boolean]("completed") ~
    get[String]("comments") ~
    get[String]("name") map {
      case id ~ completed ~ comments ~ name => Task(id, name, comments, completed)
    }
  }

  def getById(id: Long): Option[Task] = {
        DB.withConnection { implicit c =>
          SQL("select * from task where id = {id}").on('id -> id).as(task.singleOpt)
        }
  }

  def all(): List[Task] = DB.withConnection { implicit c =>
    SQL("select * from task").as(task *)
  }

  def create(task: Task) {
    DB.withConnection { implicit c =>
      SQL("""
        insert into task (name, comments, completed) values (
          {name}, {comments}, {completed}
        )
        """).bind(task).executeInsert()
    }
  }
```

FUTURE SCOPE

- Task Prioritization and Categorization
- Reminders and Notifications
- Collaboration (Multiple Users)
- Integration with other tools
- Mobile App Development


THANK YOU!