```java
@RestController
@RequestMapping("/infyinterns")
@Validated
public class ProjectAllocationAPI
{
    @Autowired
    private ProjectAllocationService projectService;

    @Autowired
    private Environment environment;

    // add new project along with mentor details
    @PostMapping("/project")
    public ResponseEntity<String> allocateProject(@Valid @RequestBody ProjectDTO project) throws InfyInternException
    {
        Integer projectId = projectService.allocateProject(project);
        String msg = environment.getProperty("API.ALLOCATION_SUCCESS")+":"+projectId;
        return new ResponseEntity<>(msg,HttpStatus.CREATED);
    }

    // get mentors based on idea owner
    @GetMapping("/mentor/{numberOfProjectsMentored}")
    public ResponseEntity<List<MentorDTO>> getMentors(@PathVariable Integer numberOfProjectsMentored) throws InfyInternException
    {
        List<MentorDTO> list = projectService.getMentors(numberOfProjectsMentored);
        return new ResponseEntity<>(list,HttpStatus.OK);
    }

    // update the mentor of a project
    @PutMapping("/project/{projectId}/{mentorId}")
    public ResponseEntity<String> updateProjectMentor(@PathVariable Integer projectId,
                                @PathVariable @Min(value=1000,message="{mentor.mentorid.invalid}")
                                @Max(value=9999,message="{mentor.mentorid.invalid}")Integer mentorId) throws InfyInternException
    {
        projectService.updateProjectMentor(projectId, mentorId);
        String msg = environment.getProperty("API.PROJECT_UPDATE_SUCCESS");
        return new ResponseEntity<>(msg,HttpStatus.OK);
    }

    // delete a project
    @DeleteMapping("/project/{projectId}")
    public ResponseEntity<String> deleteProject(@PathVariable Integer projectId) throws InfyInternException
    {
```

```java
    // add new project along with mentor details
    @PostMapping("/project")
    public ResponseEntity<String> allocateProject(@Valid @RequestBody ProjectDTO project) throws InfyInternException
    {
        Integer projectId = projectService.allocateProject(project);
        String msg = environment.getProperty("API.ALLOCATION_SUCCESS")+":"+projectId;
        return new ResponseEntity<>(msg,HttpStatus.CREATED);
    }

    // get mentors based on idea owner
    @GetMapping("/mentor/{numberOfProjectsMentored}")
    public ResponseEntity<List<MentorDTO>> getMentors(@PathVariable Integer numberOfProjectsMentored) throws InfyInternException
    {
        List<MentorDTO> list = projectService.getMentors(numberOfProjectsMentored);
        return new ResponseEntity<>(list,HttpStatus.OK);
    }

    // update the mentor of a project
    @PutMapping("/project/{projectId}/{mentorId}")
    public ResponseEntity<String> updateProjectMentor(@PathVariable Integer projectId,
                                @PathVariable @Min(value=1000,message="{mentor.mentorid.invalid}")
                                @Max(value=9999,message="{mentor.mentorid.invalid}")Integer mentorId) throws InfyInternException
    {
        projectService.updateProjectMentor(projectId, mentorId);
        String msg = environment.getProperty("API.PROJECT_UPDATE_SUCCESS");
        return new ResponseEntity<>(msg,HttpStatus.OK);
    }

    // delete a project
    @DeleteMapping("/project/{projectId}")
    public ResponseEntity<String> deleteProject(@PathVariable Integer projectId) throws InfyInternException
    {
        projectService.deleteProject(projectId);
        String msg = environment.getProperty("API.PROJECT_DELETE_SUCCESS");
        return new ResponseEntity<>(msg,HttpStatus.OK);
    }

}
```

Writable | Smart Insert | 59 : 29 : 2429

```java
package com.infy.infyinterns.dto;

import javax.validation.constraints.Max;


public class MentorDTO {
    @NotNull(message="{mentor.mentorid.absent}")
    @Min(value=1000,message="{mentor.mentorid.invalid}")
    @Max(value=9999,message="{mentor.mentorid.invalid}")
    private Integer mentorId;
    private String mentorName;
    private Integer numberOfProjectsMentored;

    public MentorDTO() {
        super();
    }

    public MentorDTO(Integer mentorId, String mentorName, Integer numberOfP
        super();
        this.mentorId = mentorId;
        this.mentorName = mentorName;
        this.numberOfProjectsMentored = numberOfProjectsMentored;
    }

    public Integer getMentorId() {
        return mentorId;
    }

    public void setMentorId(Integer mentorId) {
        this.mentorId = mentorId;
    }

    public String getMentorName() {
        return mentorName;
    }

    public void setMentorName(String mentorName) {
        this.mentorName = mentorName;
    }

    public Integer getNumberOfProjectsMentored() {
        return numberOfProjectsMentored;
    }

    public void setNumberOfProjectsMentored(Integer numberOfProjectsMentored
```

ProjectDTO.java

```java
package com.infy.infyinterns.dto;

import java.time.LocalDate;


public class ProjectDTO {

    private Integer projectId;
    @NotNull(message="{project.projectname.absent}")
    private String projectName;
    @NotNull(message="{project.ideaowner.absent}")
    private Integer ideaOwner;
    @NotNull(message="{project.releasedate.absent}")
    private LocalDate releaseDate;
    @NotNull(message="{project.mentor.absent}")
    @Valid
    private MentorDTO mentorDTO;

    public ProjectDTO() {
        super();
    }

    public ProjectDTO(Integer projectId, String projectName,
            Integer ideaOwner, LocalDate releaseDate,
            MentorDTO mentorDTO) {
        super();
        this.projectId = projectId;
        this.projectName = projectName;
        this.ideaOwner = ideaOwner;
        this.releaseDate = releaseDate;
        this.mentorDTO = mentorDTO;
    }

    public Integer getProjectId() {
        return projectId;
    }

    public void setProjectId(Integer projectId) {
        this.projectId = projectId;
    }

    public String getProjectName() {
        return projectName;
    }
```

Mentor.java

```java
package com.infy.infyinterns.entity;

import javax.persistence.Column;
import javax.persistence.Entity;
import javax.persistence.Id;

@Entity
public class Mentor
{
    @Id
    private Integer mentorId;
    private String mentorName;
    @Column(name="projects_mentored")
    private Integer numberOfProjectsMentored;

    public Integer getMentorId()
    {
    return mentorId;
    }

    public void setMentorId(Integer mentorId)
    {
    this.mentorId = mentorId;
    }

    public String getMentorName()
    {
    return mentorName;
    }

    public void setMentorName(String mentorName)
    {
    this.mentorName = mentorName;
    }

    public Integer getNumberOfProjectsMentored()
    {
```

Project.java

```java
package com.infy.infyinterns.entity;

import java.time.LocalDate;

import javax.persistence.CascadeType;
import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;
import javax.persistence.JoinColumn;
import javax.persistence.ManyToOne;

@Entity
public class Project
{
    @Id
    @GeneratedValue(strategy=GenerationType.IDENTITY)
    private Integer projectId;
    private String projectName;
    private Integer ideaOwner;
    private LocalDate releaseDate;
```

Window Help

MentorRepository.java

```java
package com.infy.infyinterns.repository;

import java.util.List;

public interface MentorRepository extends CrudRepository<Mentor, Integer>
{
    // add methods if required
    List<Mentor> findByNumberOfProjectsMentored(Integer numberOfProjectsMentored);
}

```

ProjectRepository.java

```java
package com.infy.infyinterns.repository;

import org.springframework.data.repository.CrudRepository;

import com.infy.infyinterns.entity.Project;

public interface ProjectRepository extends CrudRepository<Project, Integer>
{

    // add methods if required

}

```

Window Help

ProjectAllocationServiceImpl.java

```java
import java.util.List;

@Service(value="projectService")
@Transactional
public class ProjectAllocationServiceImpl implements ProjectAllocationService {

    @Autowired
    private ProjectRepository projectRepository;
    @Autowired
    private MentorRepository mentorRepository;
    @Override
    public Integer allocateProject(ProjectDTO project) throws InfyInternException {
        Optional<Mentor> optional = mentorRepository.findById(project.getMentorDTO().getMentorId());
        Mentor mentor = optional.orElseThrow(() -> new InfyInternException("Service.MENTOR_NOT_FOUND"));
        if(mentor.getNumberOfProjectsMentored()>=3)
            throw new InfyInternException("Service.CANNOT_ALLOCATE_PROJECT");
        Project project1 = new Project();
        project1.setProjectId(project.getProjectId());
        project1.setProjectName(project.getProjectName());
        project1.setIdeaOwner(project.getIdeaOwner());
        project1.setReleaseDate(project.getReleaseDate());
        project1.setMentor(mentor);
        mentor.setNumberOfProjectsMentored(mentor.getNumberOfProjectsMentored()+1);
        Project project2 = projectRepository.save(project1);

        return project2.getProjectId();
    }


    @Override
    public List<MentorDTO> getMentors(Integer numberOfProjectsMentored) throws InfyInternException {
        List<Mentor> list1 = mentorRepository.findByNumberOfProjectsMentored(numberOfProjectsMentored);
        if(list1.isEmpty())
            throw new InfyInternException("Service.MENTOR_NOT_FOUND");
        List<MentorDTO> list2 = list1.stream().map(x -> new MentorDTO(x.getMentorId(),x.getMentorName(),x.getM
```

ProjectAllocationServiceImpl.java

```java
        project1.setMentor(mentor);
        mentor.setNumberOfProjectsMentored(mentor.getNumberOfProjectsMentored()+1);
        Project project2 = projectRepository.save(project1);

        return project2.getProjectId();
    }


    @Override
    public List<MentorDTO> getMentors(Integer numberOfProjectsMentored) throws InfyInternException {
        List<Mentor> list1 = mentorRepository.findByNumberOfProjectsMentored(numberOfProjectsMentored);
        if(list1.isEmpty())
            throw new InfyInternException("Service.MENTOR_NOT_FOUND");
        List<MentorDTO> list2 = list1.stream().map(x -> new MentorDTO(x.getMentorId(),x.getMentorName(),x.getNumberOfProjectsMentored()))
                                                .collect(Collectors.toList());

        return list2;
    }


    @Override
    public void updateProjectMentor(Integer projectId, Integer mentorId) throws InfyInternException {
        Optional<Mentor> optional = mentorRepository.findById(mentorId);
        Mentor mentor = optional.orElseThrow(() -> new InfyInternException("Service.MENTOR_NOT_FOUND"));
        if(mentor.getNumberOfProjectsMentored()>=3)
            throw new InfyInternException("Service.CANNOT_ALLOCATE_PROJECT");
        Optional<Project> optional1 = projectRepository.findById(projectId);
        Project project = optional1.orElseThrow(() -> new InfyInternException("Service.PROJECT_NOT_FOUND"));
        project.setMentor(mentor);
        mentor.setNumberOfProjectsMentored(mentor.getNumberOfProjectsMentored()+1);


    }

    @Override
    public void deleteProject(Integer projectId) throws InfyInternException {
        Optional<Project> optional1 = projectRepository.findById(projectId);
        Project project = optional1.orElseThrow(() -> new InfyInternException("Service.PROJECT_NOT_FOUND"));
        if(project.getMentor()==null)
            projectRepository.delete(project);
        else
        {
            Mentor mentor = project.getMentor();
            mentor.setNumberOfProjectsMentored(mentor.getNumberOfProjectsMentored()-1);
            project.setMentor(null);
```

Writable Smart Insert 1 : 1 : 0

```java
        return list2;
    }


    @Override
    public void updateProjectMentor(Integer projectId, Integer mentorId) throws InfyInternException {
        Optional<Mentor> optional = mentorRepository.findById(mentorId);
        Mentor mentor = optional.orElseThrow(() -> new InfyInternException("Service.MENTOR_NOT_FOUND"));
        if(mentor.getNumberOfProjectsMentored()>=3)
            throw new InfyInternException("Service.CANNOT_ALLOCATE_PROJECT");
        Optional<Project> optional1 = projectRepository.findById(projectId);
        Project project = optional1.orElseThrow(() -> new InfyInternException("Service.PROJECT_NOT_FOUND"));
        project.setMentor(mentor);
        mentor.setNumberOfProjectsMentored(mentor.getNumberOfProjectsMentored()+1);


    }

    @Override
    public void deleteProject(Integer projectId) throws InfyInternException {
        Optional<Project> optional1 = projectRepository.findById(projectId);
        Project project = optional1.orElseThrow(() -> new InfyInternException("Service.PROJECT_NOT_FOUND"));
        if(project.getMentor()==null)
            projectRepository.delete(project);
        else
        {
            Mentor mentor = project.getMentor();
            mentor.setNumberOfProjectsMentored(mentor.getNumberOfProjectsMentored()-1);
            project.setMentor(null);
            projectRepository.delete(project);
        }
    }
}
```

Writable | Smart Insert | 1 : 1 : 0

ASUS VivoBook

ExceptionControllerAdvice.java

```java
package com.infy.infyinterns.utility;

import java.util.stream.Collectors;
@RestControllerAdvice
public class ExceptionControllerAdvice
{

    private static final Log LOGGER = LogFactory.getLog(ExceptionControllerAdvice.class);
    @Autowired
    private Environment environment;

    // add appropriate annotation
    @ExceptionHandler(InfyInternException.class)
    public ResponseEntity<ErrorInfo> infyInternExceptionHandler(InfyInternException exception)
    {
    LOGGER.error(exception.getMessage(), exception);
    ErrorInfo errorInfo = new ErrorInfo();
    errorInfo.setErrorCode(HttpStatus.BAD_REQUEST.value());
    errorInfo.setErrorMessage(environment.getProperty(exception.getMessage()));
    return new ResponseEntity<>(errorInfo, HttpStatus.BAD_REQUEST);
    }

    // add appropriate annotation
    @ExceptionHandler(Exception.class)
    public ResponseEntity<ErrorInfo> generalExceptionHandler(Exception exception)
    {
    LOGGER.error(exception.getMessage(), exception);
    ErrorInfo errorInfo = new ErrorInfo();
    errorInfo.setErrorCode(HttpStatus.INTERNAL_SERVER_ERROR.value());
    errorInfo.setErrorMessage(environment.getProperty("General.EXCEPTION_MESSAGE"));
    return new ResponseEntity<>(errorInfo,
                    HttpStatus.INTERNAL_SERVER_ERROR);
    }
```

```java
    // add appropriate annotation
    @ExceptionHandler(InfyInternException.class)
    public ResponseEntity<ErrorInfo> infyInternExceptionHandler(InfyInternException exception)
    {
    LOGGER.error(exception.getMessage(), exception);
    ErrorInfo errorInfo = new ErrorInfo();
    errorInfo.setErrorCode(HttpStatus.BAD_REQUEST.value());
    errorInfo.setErrorMessage(environment.getProperty(exception.getMessage()));
    return new ResponseEntity<>(errorInfo, HttpStatus.BAD_REQUEST);
    }

    // add appropriate annotation
    @ExceptionHandler(Exception.class)
    public ResponseEntity<ErrorInfo> generalExceptionHandler(Exception exception)
    {
    LOGGER.error(exception.getMessage(), exception);
    ErrorInfo errorInfo = new ErrorInfo();
    errorInfo.setErrorCode(HttpStatus.INTERNAL_SERVER_ERROR.value());
    errorInfo.setErrorMessage(environment.getProperty("General.EXCEPTION_MESSAGE"));
    return new ResponseEntity<>(errorInfo,
                    HttpStatus.INTERNAL_SERVER_ERROR);
    }

    // add appropriate annotation
    @ExceptionHandler({MethodArgumentNotValidException.class,ConstraintViolationException.class})
    public ResponseEntity<ErrorInfo> validatorExceptionHandler(Exception exception)
    {
    LOGGER.error(exception.getMessage(), exception);
    String errorMsg;
    if (exception instanceof MethodArgumentNotValidException)
    {
        MethodArgumentNotValidException manvException = (MethodArgumentNotValidException) exception;
        errorMsg = manvException.getBindingResult()
                    .getAllErrors()
```

Writable | Smart Insert | 1:1:0

aspect.java - Eclipse IDE

Window Help

Quick Access

ExceptionControllerAdvice.java | LoggingAspect.java

```java
package com.infy.infyinterns.utility;

import org.apache.commons.logging.Log;
@Component
@Aspect
public class LoggingAspect
{

    private static final Log LOGGER = LogFactory.getLog(LoggingAspect.class);
    @AfterThrowing(pointcut="execution(* com.infy.infyinterns.service.ProjectAllocationServiceImpl.*(..))", throwing="exception")
    public void logServiceException(InfyInternException exception)
    {
    // code
        LOGGER.error(exception.getMessage(),exception);
    }

}
```

InfyInternsApplicationTests.java

```java
package com.infy.infyinterns;

import java.time.LocalDate;...

@SpringBootTest
public class InfyInternsApplicationTests {

    @Mock
    private MentorRepository mentorRepository;

    @InjectMocks
    private ProjectAllocationService projectAllocationService = new ProjectAllocationServiceImpl();

    @Test
    public void allocateProjectCannotAllocateTest() throws Exception {
        MentorDTO mentorDTO = new MentorDTO(100,"Warner",6);
        ProjectDTO projectDTO = new ProjectDTO(1,"Shoe Cart",10012,LocalDate.of(2021, 06, 02),mentorDTO);
        Mentor mentor = new Mentor();
        mentor.setMentorId(mentorDTO.getMentorId());
        mentor.setMentorName(mentorDTO.getMentorName());
        mentor.setNumberOfProjectsMentored(mentorDTO.getNumberOfProjectsMentored());
        Mockito.when(mentorRepository.findById(mentorDTO.getMentorId())).thenReturn(Optional.of(mentor));
        InfyInternException exception = Assertions.assertThrows(InfyInternException.class, () -> projectAllocati
        Assertions.assertEquals("Service.CANNOT_ALLOCATE_PROJECT", exception.getMessage());


    }

    @Test
    public void allocateProjectMentorNotFoundTest() throws Exception {
        MentorDTO mentorDTO = new MentorDTO(1019,"Bethany",2);
        ProjectDTO projectDTO = new ProjectDTO(1,"Shoe Cart",10012,LocalDate.of(2021, 06, 02),mentorDTO);

        Mockito.when(mentorRepository.findById(projectDTO.getMentorDTO().getMentorId())).thenReturn(Optional.emp
        InfyInternException exception = Assertions.assertThrows(InfyInternException.class, () -> projectAllocati
        Assertions.assertEquals("Service.MENTOR_NOT_FOUND", exception.getMessage());
```

```
sts {

pository;

rojectAllocationService = new ProjectAllocationServiceImpl();

llocateTest() throws Exception {
orDTO(100,"Warner",6);
ojectDTO(1,"Shoe Cart",10012,LocalDate.of(2021, 06, 02),mentorDTO);

etMentorId());
getMentorName());
ored(mentorDTO.getNumberOfProjectsMentored());
findById(mentorDTO.getMentorId())).thenReturn(Optional.of(mentor));
Assertions.assertThrows(InfyInternException.class, () -> projectAllocationService.allocateProject(projectDTO));
e.CANNOT_ALLOCATE_PROJECT", exception.getMessage());


tFoundTest() throws Exception {
rDTO(1019,"Bethany",2);
jectDTO(1,"Shoe Cart",10012,LocalDate.of(2021, 06, 02),mentorDTO);

ndById(projectDTO.getMentorDTO().getMentorId())).thenReturn(Optional.empty());
Assertions.assertThrows(InfyInternException.class, () -> projectAllocationService.allocateProject(projectDTO));
.MENTOR_NOT_FOUND", exception.getMessage());
```

```java
public class InfyInternsApplicationTests {

    @Mock
    private MentorRepository mentorRepository;

    @InjectMocks
    private ProjectAllocationService projectAllocationService = new ProjectAllocationServiceImpl();

    @Test
    public void allocateProjectCannotAllocateTest() throws Exception {
        MentorDTO mentorDTO = new MentorDTO(100,"Warner",6);
        ProjectDTO projectDTO = new ProjectDTO(1,"Shoe Cart",10012,LocalDate.of(2021, 06, 02),mentorDTO);
        Mentor mentor = new Mentor();
        mentor.setMentorId(mentorDTO.getMentorId());
        mentor.setMentorName(mentorDTO.getMentorName());
        mentor.setNumberOfProjectsMentored(mentorDTO.getNumberOfProjectsMentored());
        Mockito.when(mentorRepository.findById(mentorDTO.getMentorId())).thenReturn(Optional.of(mentor));
        InfyInternException exception = Assertions.assertThrows(InfyInternException.class, () -> projectAllocationService.allocateProject(projectDTO));
        Assertions.assertEquals("Service.CANNOT_ALLOCATE_PROJECT", exception.getMessage());


    }

    @Test
    public void allocateProjectMentorNotFoundTest() throws Exception {
        MentorDTO mentorDTO = new MentorDTO(1019,"Bethany",2);
        ProjectDTO projectDTO = new ProjectDTO(1,"Shoe Cart",10012,LocalDate.of(2021, 06, 02),mentorDTO);

        Mockito.when(mentorRepository.findById(projectDTO.getMentorDTO().getMentorId())).thenReturn(Optional.empty());
        InfyInternException exception = Assertions.assertThrows(InfyInternException.class, () -> projectAllocationService.allocateProject(projectDTO));
        Assertions.assertEquals("Service.MENTOR_NOT_FOUND", exception.getMessage());
    }
}
```

Writable | Smart Insert | 1:1:0

ENG IN 16:29 23-04-2021

ASUS VivoBook

Verifying Your Code

100%	0%	100%	100%
Total Passed Test Cases	Total Failed Test Cases	Structural Correctness	Logical Correctness

Overall Report | Detailed Report

Structural Test Cases

Test Group	Test Class	Total Test Cases	Passed Test Cases	Failed Test Cases
API	ProjectAllocationAPI	22	22	0
DTO	MentorDTO	12	12	0
DTO	ProjectDTO	18	18	0
Entity	Mentor	11	11	0