PROMINEO TECH

Final Project Work

Points possible: 100

Category	Criteria	% of Grade
Functionality	Does the code work?	25
Organization	Is the code clean and organized? Proper use of white space, syntax, and consistency are utilized. Names and comments are concise and clear.	25
Creativity	Student solved the problems presented in the assignment using creativity and out of the box thinking.	25
Completeness	All requirements of the assignment are complete that were committed to for the given week.	25

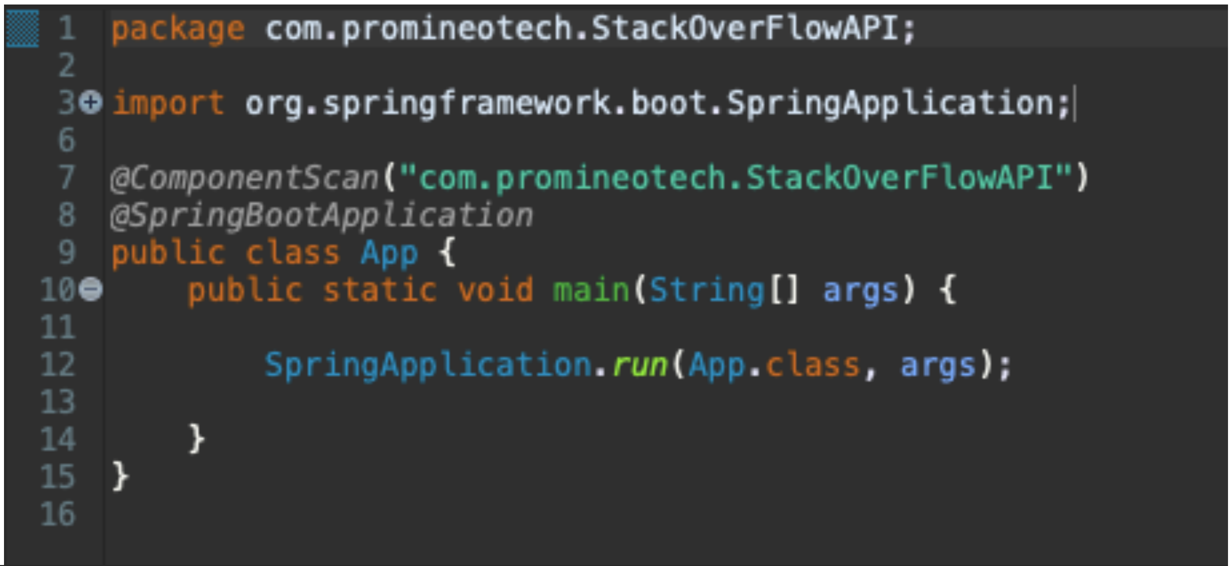
Instructions: In Eclipse, or an IDE of your choice, write the code that accomplishes the objectives listed below. Ensure that the code compiles and runs as directed. Take screenshots of the code and of the running program (make sure to get screenshots of all required functionality) and paste them in this document where instructed below. Create a new repository on GitHub for this week's assignments and push this document, with your Java project code, to the repository. Add the URL for this week's repository to this document where instructed and submit this document to your instructor when complete.

Coding Steps:

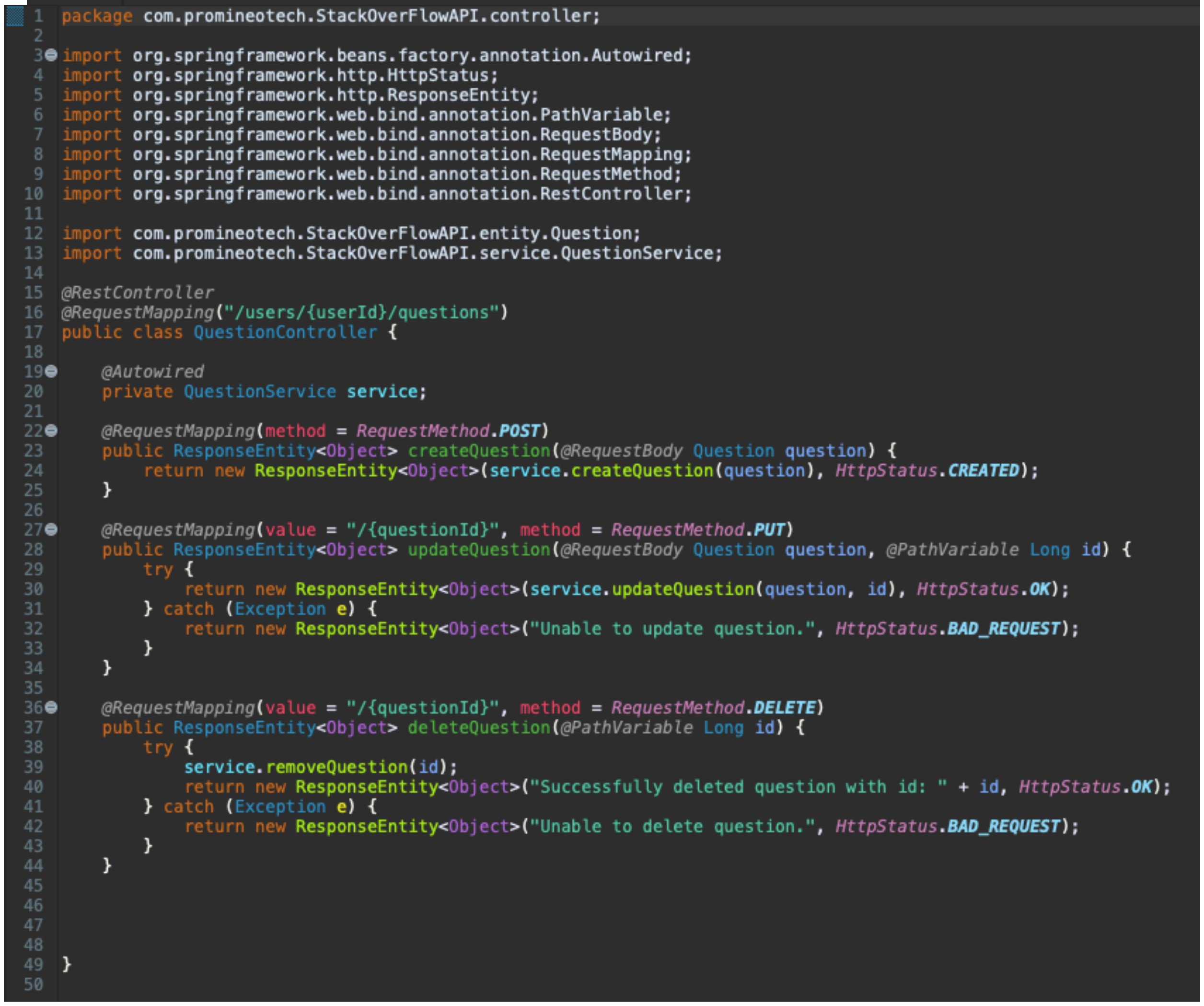
Continue contributing to your final project.

Screenshots of Code:

PROMINEO TECH

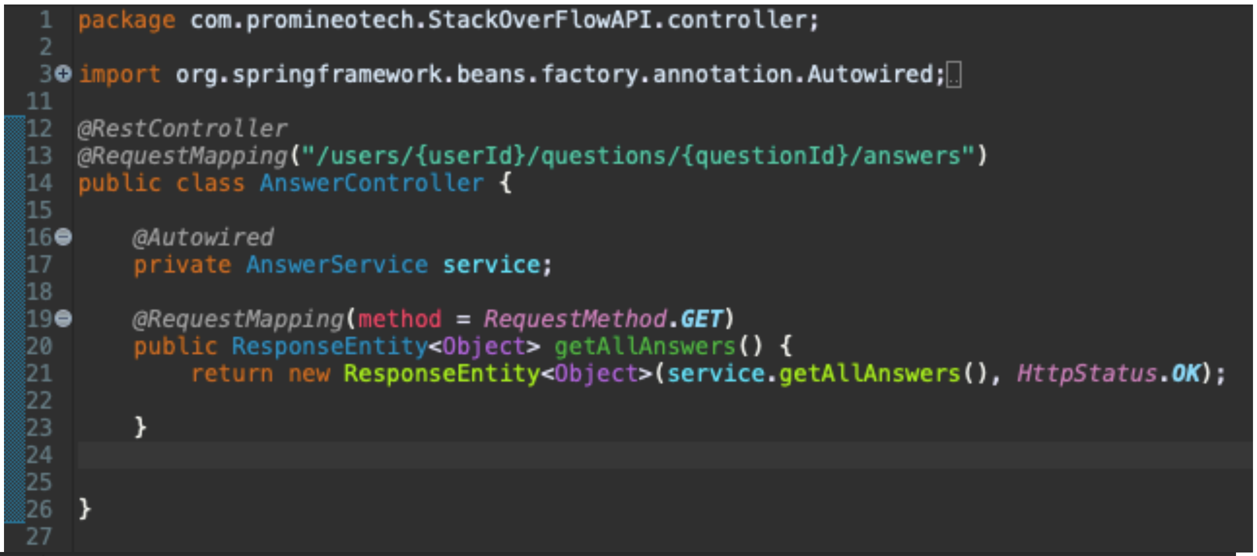


```java
package com.promineotech.StackOverFlowAPI;

import org.springframework.boot.SpringApplication;

@ComponentScan("com.promineotech.StackOverFlowAPI")
@SpringBootApplication
public class App {
    public static void main(String[] args) {

        SpringApplication.run(App.class, args);

    }
}

```

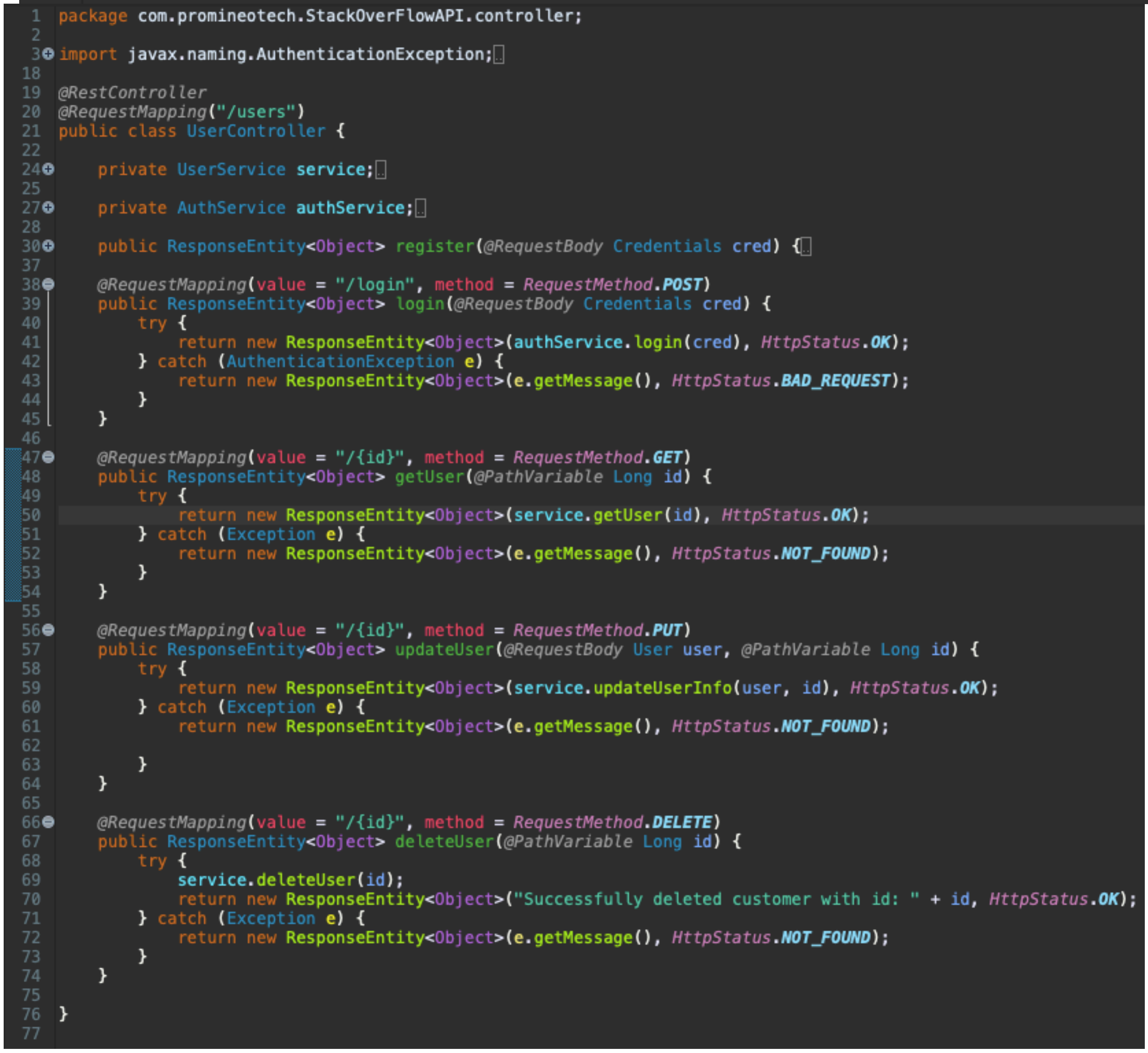


```java
package com.promineotech.StackOverFlowAPI.controller;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.http.HttpStatus;
import org.springframework.http.ResponseEntity;
import org.springframework.web.bind.annotation.PathVariable;
import org.springframework.web.bind.annotation.RequestBody;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestMethod;
import org.springframework.web.bind.annotation.RestController;

import com.promineotech.StackOverFlowAPI.entity.Question;
import com.promineotech.StackOverFlowAPI.service.QuestionService;

@RestController
@RequestMapping("/users/{userId}/questions")
public class QuestionController {

    @Autowired
    private QuestionService service;

    @RequestMapping(method = RequestMethod.POST)
    public ResponseEntity<Object> createQuestion(@RequestBody Question question) {
        return new ResponseEntity<Object>(service.createQuestion(question), HttpStatus.CREATED);
    }

    @RequestMapping(value = "/{questionId}", method = RequestMethod.PUT)
    public ResponseEntity<Object> updateQuestion(@RequestBody Question question, @PathVariable Long id) {
        try {
            return new ResponseEntity<Object>(service.updateQuestion(question, id), HttpStatus.OK);
        } catch (Exception e) {
            return new ResponseEntity<Object>("Unable to update question.", HttpStatus.BAD_REQUEST);
        }
    }

    @RequestMapping(value = "/{questionId}", method = RequestMethod.DELETE)
    public ResponseEntity<Object> deleteQuestion(@PathVariable Long id) {
        try {
            service.removeQuestion(id);
            return new ResponseEntity<Object>("Successfully deleted question with id: " + id, HttpStatus.OK);
        } catch (Exception e) {
            return new ResponseEntity<Object>("Unable to delete question.", HttpStatus.BAD_REQUEST);
        }
    }




}

```

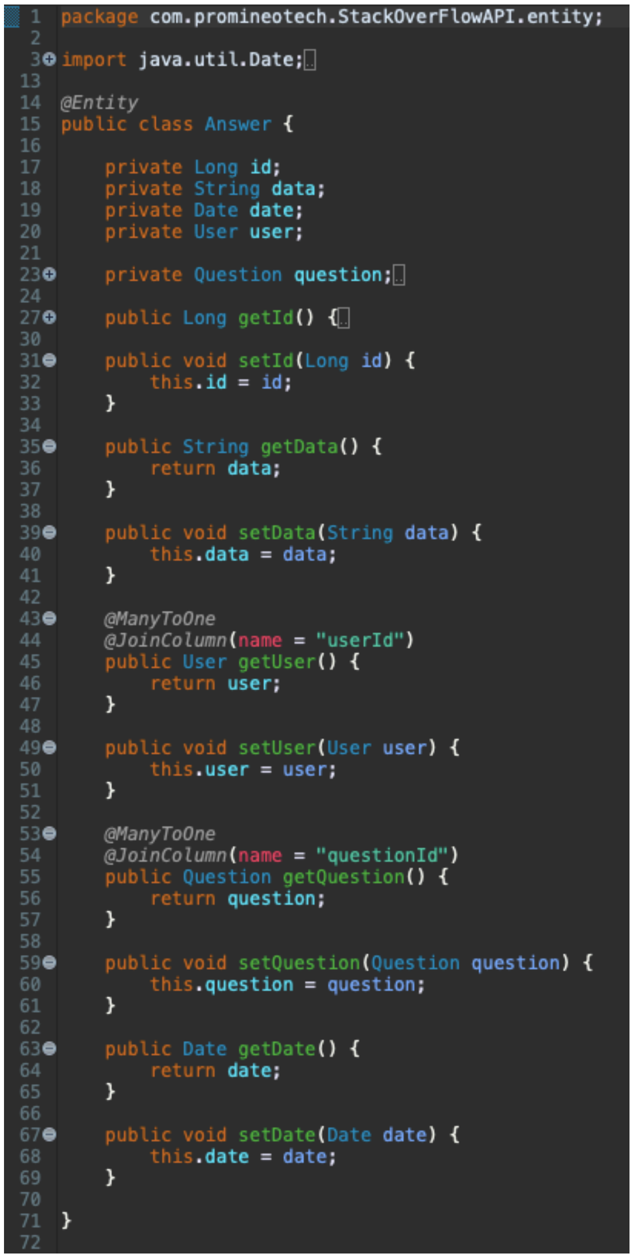
PROMINEO TECH



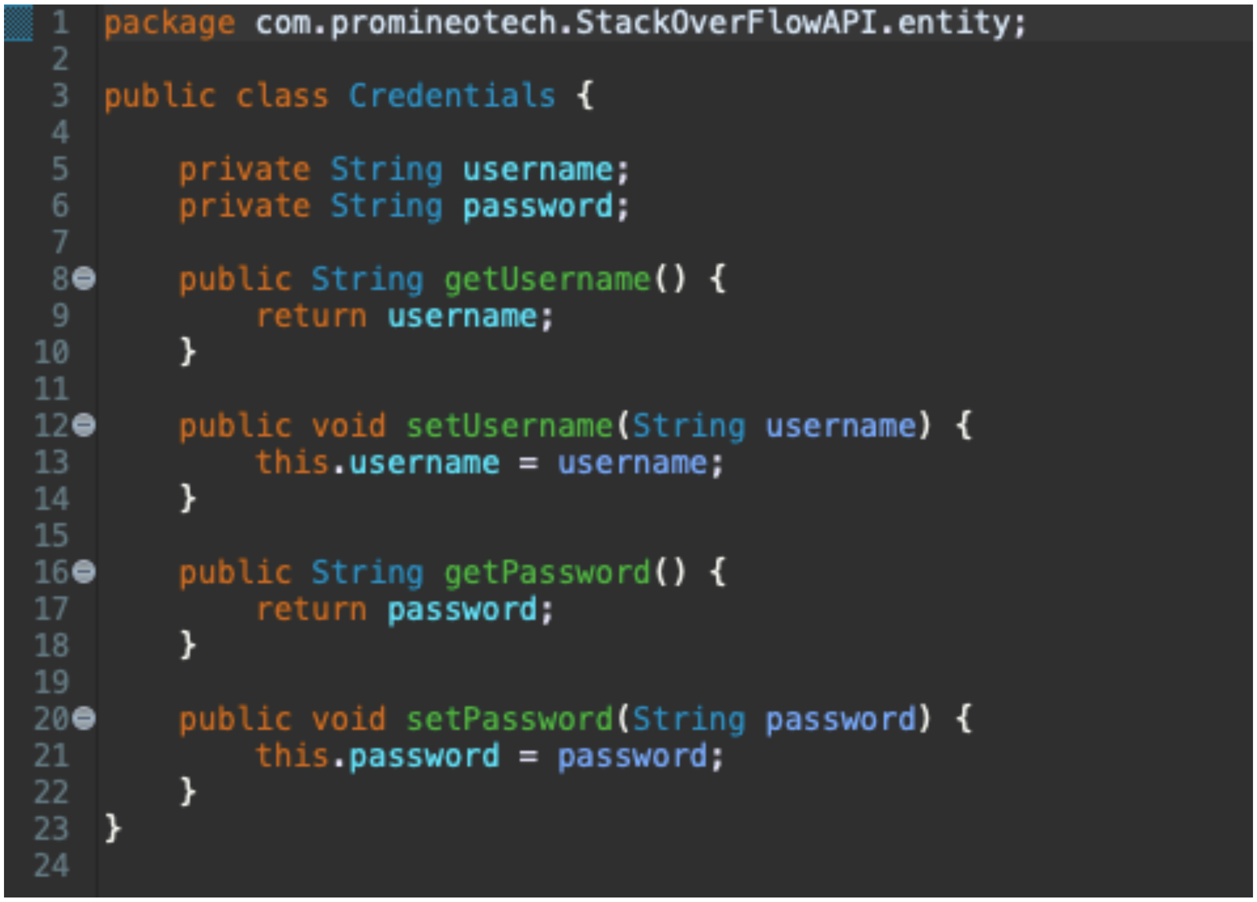
```java
package com.promineotech.StackOverFlowAPI.controller;

import org.springframework.beans.factory.annotation.Autowired;

@RestController
@RequestMapping("/users/{userId}/questions/{questionId}/answers")
public class AnswerController {

    @Autowired
    private AnswerService service;

    @RequestMapping(method = RequestMethod.GET)
    public ResponseEntity<Object> getAllAnswers() {
        return new ResponseEntity<Object>(service.getAllAnswers(), HttpStatus.OK);

    }


}
```



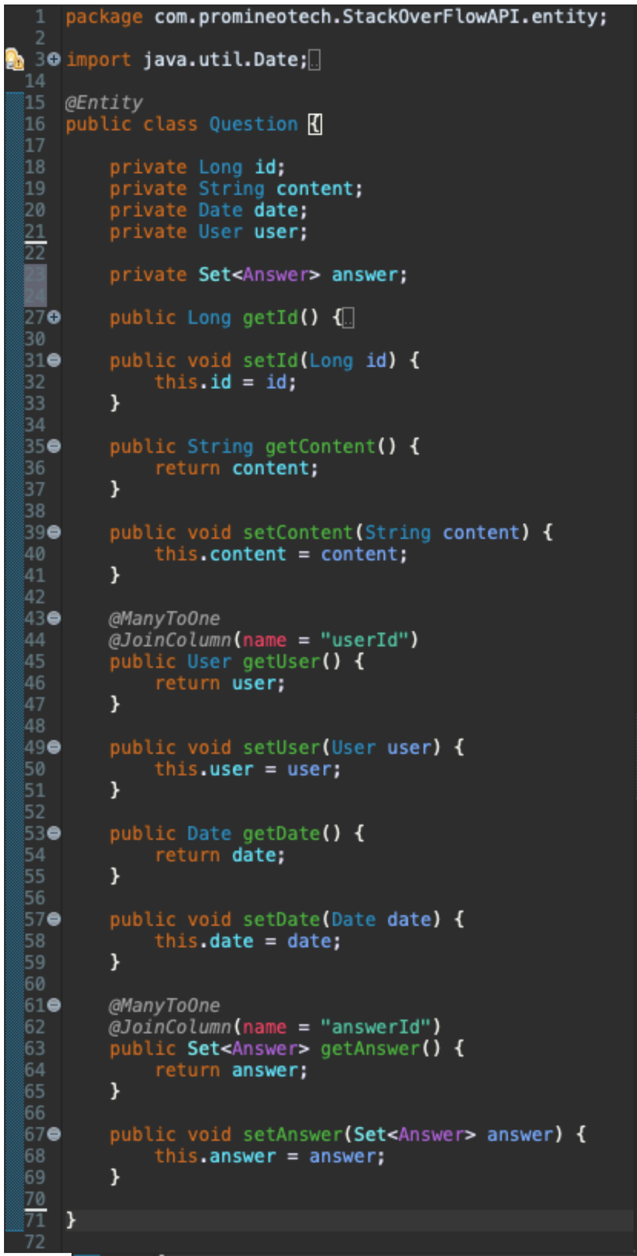
```java
package com.promineotech.StackOverFlowAPI.controller;

import javax.naming.AuthenticationException;

@RestController
@RequestMapping("/users")
public class UserController {

    private UserService service;

    private AuthService authService;

    public ResponseEntity<Object> register(@RequestBody Credentials cred) {

    @RequestMapping(value = "/login", method = RequestMethod.POST)
    public ResponseEntity<Object> login(@RequestBody Credentials cred) {
        try {
            return new ResponseEntity<Object>(authService.login(cred), HttpStatus.OK);
        } catch (AuthenticationException e) {
            return new ResponseEntity<Object>(e.getMessage(), HttpStatus.BAD_REQUEST);
        }
    }

    @RequestMapping(value = "/{id}", method = RequestMethod.GET)
    public ResponseEntity<Object> getUser(@PathVariable Long id) {
        try {
            return new ResponseEntity<Object>(service.getUser(id), HttpStatus.OK);
        } catch (Exception e) {
            return new ResponseEntity<Object>(e.getMessage(), HttpStatus.NOT_FOUND);
        }
    }

    @RequestMapping(value = "/{id}", method = RequestMethod.PUT)
    public ResponseEntity<Object> updateUser(@RequestBody User user, @PathVariable Long id) {
        try {
            return new ResponseEntity<Object>(service.updateUserInfo(user, id), HttpStatus.OK);
        } catch (Exception e) {
            return new ResponseEntity<Object>(e.getMessage(), HttpStatus.NOT_FOUND);

        }
    }

    @RequestMapping(value = "/{id}", method = RequestMethod.DELETE)
    public ResponseEntity<Object> deleteUser(@PathVariable Long id) {
        try {
            service.deleteUser(id);
            return new ResponseEntity<Object>("Successfully deleted customer with id: " + id, HttpStatus.OK);
        } catch (Exception e) {
            return new ResponseEntity<Object>(e.getMessage(), HttpStatus.NOT_FOUND);
        }
    }

}
```



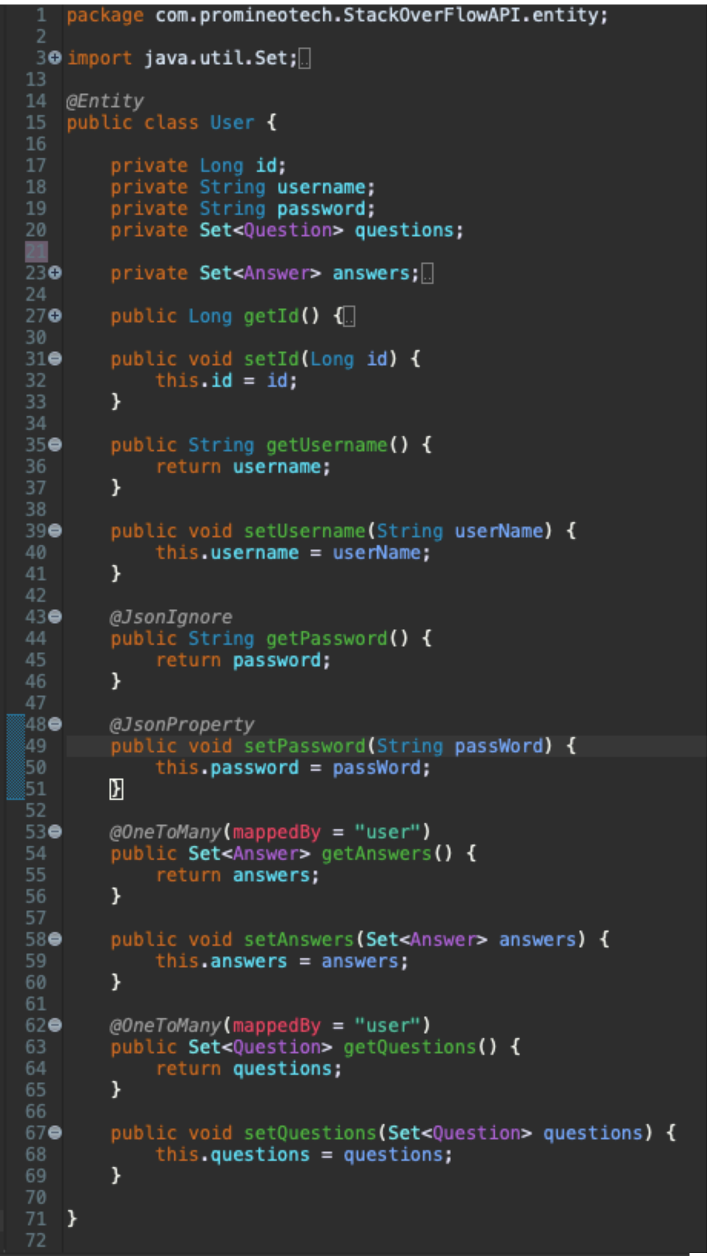
```java
package com.promineotech.StackOverFlowAPI.entity;

import java.util.Date;

@Entity
public class Answer {

    private Long id;
    private String data;
    private Date date;
    private User user;

    private Question question;

    public Long getId() {

    public void setId(Long id) {
        this.id = id;
    }

    public String getData() {
        return data;
    }

    public void setData(String data) {
        this.data = data;
    }

    @ManyToOne
    @JoinColumn(name = "userId")
    public User getUser() {
        return user;
    }

    public void setUser(User user) {
        this.user = user;
    }

    @ManyToOne
    @JoinColumn(name = "questionId")
    public Question getQuestion() {
        return question;
    }

    public void setQuestion(Question question) {
        this.question = question;
    }

    public Date getDate() {
        return date;
    }

    public void setDate(Date date) {
        this.date = date;
    }

}
```



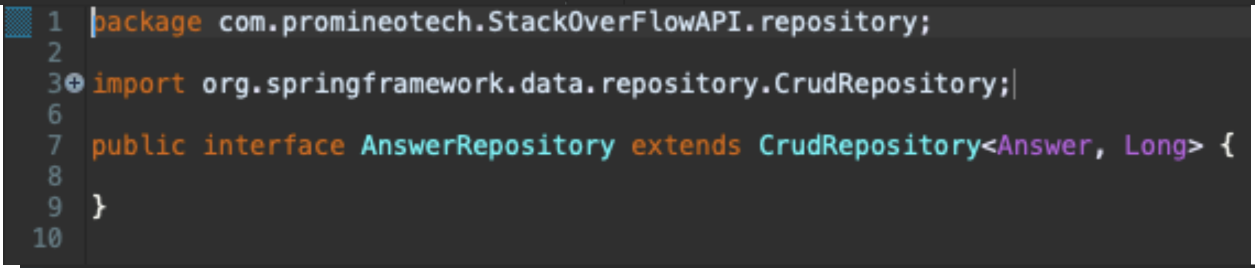
```java
package com.promineotech.StackOverFlowAPI.entity;

public class Credentials {

    private String username;
    private String password;

    public String getUsername() {
        return username;
    }

    public void setUsername(String username) {
        this.username = username;
    }

    public String getPassword() {
        return password;
    }

    public void setPassword(String password) {
        this.password = password;
    }
}

```



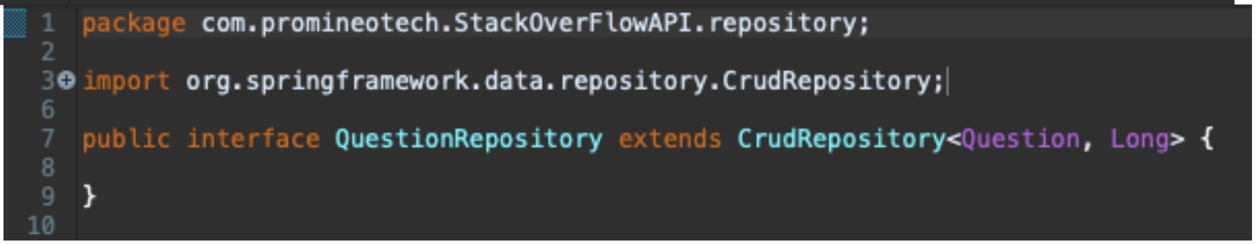
```java
package com.promineotech.StackOverFlowAPI.entity;

import java.util.Date;

@Entity
public class Question {

    private Long id;
    private String content;
    private Date date;
    private User user;

    private Set<Answer> answer;

    public Long getId() {

    public void setId(Long id) {
        this.id = id;
    }

    public String getContent() {
        return content;
    }

    public void setContent(String content) {
        this.content = content;
    }

    @ManyToOne
    @JoinColumn(name = "userId")
    public User getUser() {
        return user;
    }

    public void setUser(User user) {
        this.user = user;
    }

    public Date getDate() {
        return date;
    }

    public void setDate(Date date) {
        this.date = date;
    }

    @ManyToOne
    @JoinColumn(name = "answerId")
    public Set<Answer> getAnswer() {
        return answer;
    }

    public void setAnswer(Set<Answer> answer) {
        this.answer = answer;
    }

}
```



```java
package com.promineotech.StackOverFlowAPI.entity;

import java.util.Set;

@Entity
public class User {

    private Long id;
    private String username;
    private String password;
    private Set<Question> questions;

    private Set<Answer> answers;

    public Long getId() {

    public void setId(Long id) {
        this.id = id;
    }

    public String getUsername() {
        return username;
    }

    public void setUsername(String userName) {
        this.username = userName;
    }

    @JsonIgnore
    public String getPassword() {
        return password;
    }

    @JsonProperty
    public void setPassword(String passWord) {
        this.password = passWord;
    }

    @OneToMany(mappedBy = "user")
    public Set<Answer> getAnswers() {
        return answers;
    }

    public void setAnswers(Set<Answer> answers) {
        this.answers = answers;
    }

    @OneToMany(mappedBy = "user")
    public Set<Question> getQuestions() {
        return questions;
    }

    public void setQuestions(Set<Question> questions) {
        this.questions = questions;
    }
}
```

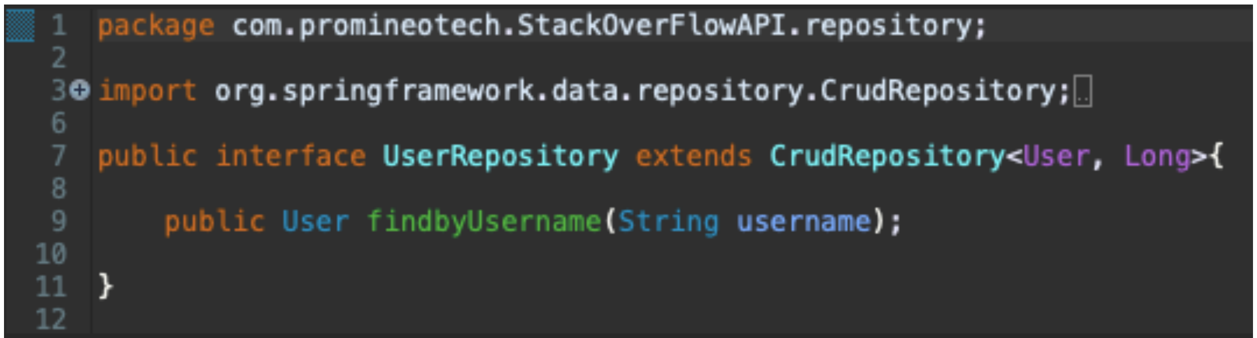


```java
package com.promineotech.StackOverFlowAPI.repository;

import org.springframework.data.repository.CrudRepository;

public interface AnswerRepository extends CrudRepository<Answer, Long> {

}
```

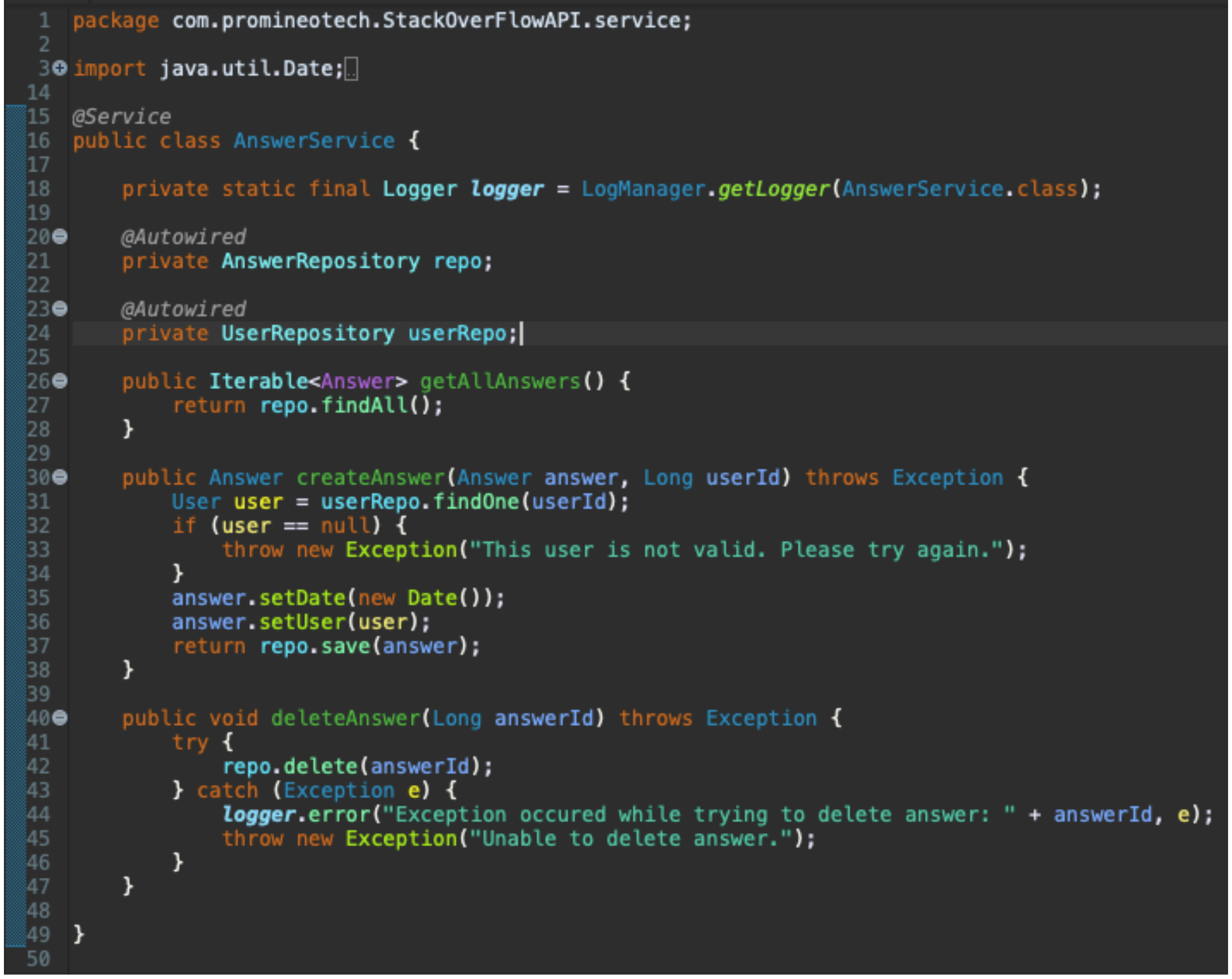


```java
package com.promineotech.StackOverFlowAPI.repository;

import org.springframework.data.repository.CrudRepository;

public interface QuestionRepository extends CrudRepository<Question, Long> {

}
```

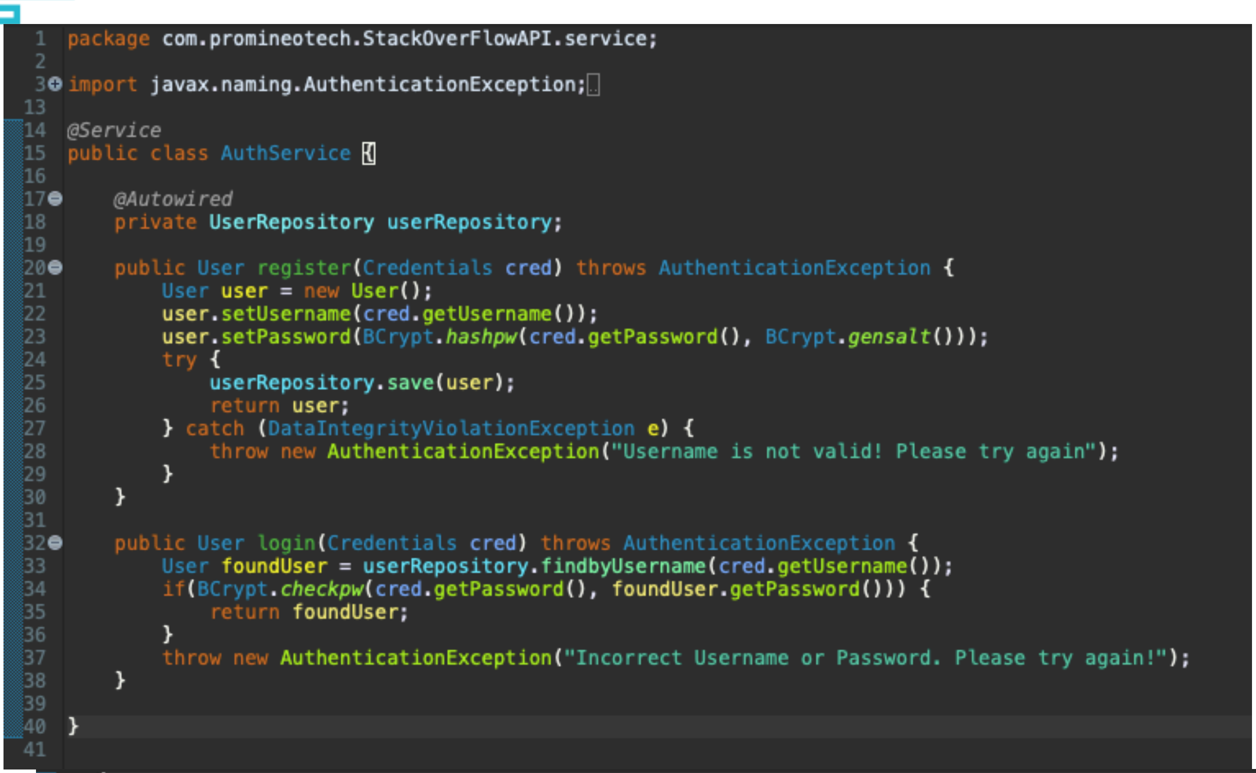
PROMINEO TECH



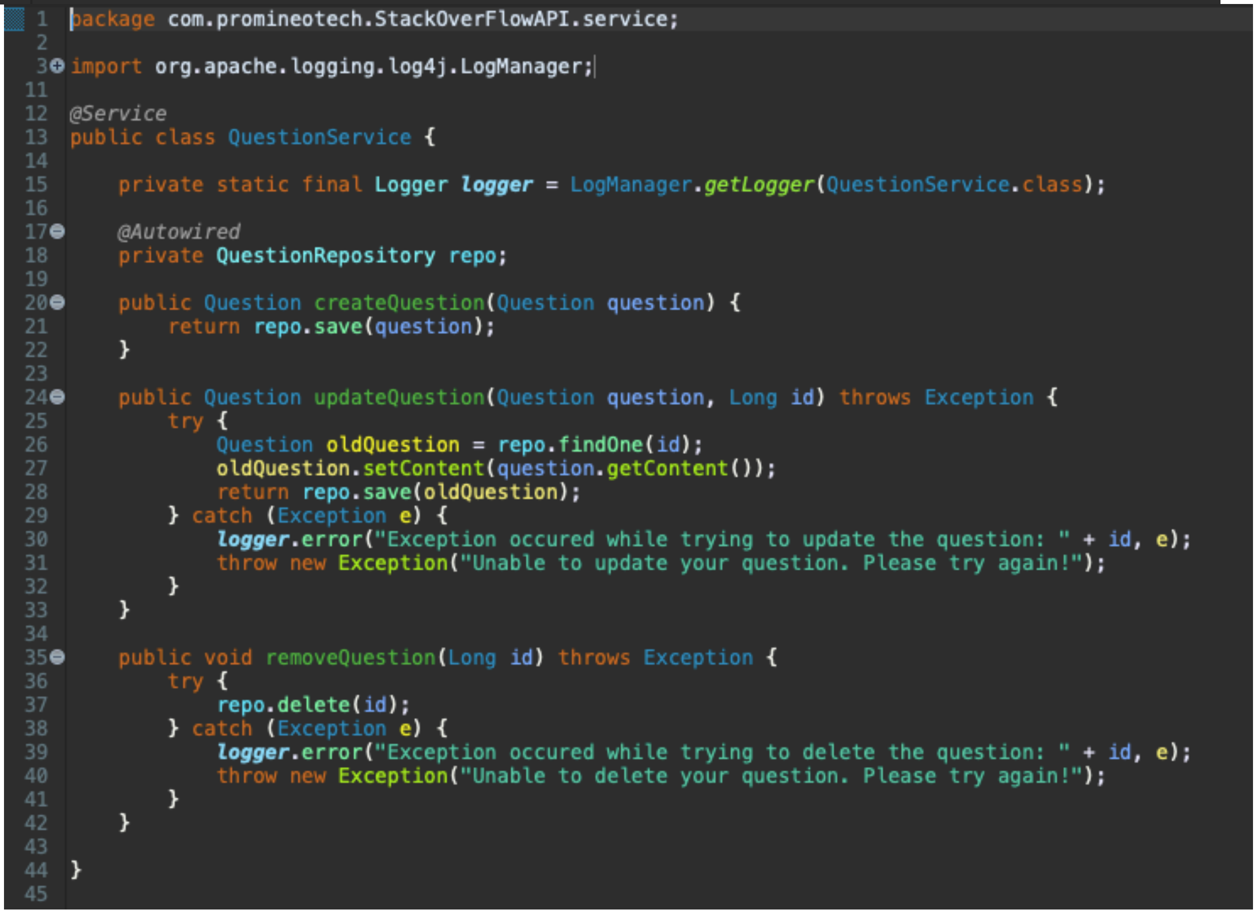
```java
package com.promineotech.StackOverFlowAPI.repository;

import org.springframework.data.repository.CrudRepository;

public interface UserRepository extends CrudRepository<User, Long>{

    public User findbyUsername(String username);

}
```



```java
package com.promineotech.StackOverFlowAPI.service;

import java.util.Date;

@Service
public class AnswerService {

    private static final Logger logger = LogManager.getLogger(AnswerService.class);

    @Autowired
    private AnswerRepository repo;

    @Autowired
    private UserRepository userRepo;

    public Iterable<Answer> getAllAnswers() {
        return repo.findAll();
    }

    public Answer createAnswer(Answer answer, Long userId) throws Exception {
        User user = userRepo.findOne(userId);
        if (user == null) {
            throw new Exception("This user is not valid. Please try again.");
        }
        answer.setDate(new Date());
        answer.setUser(user);
        return repo.save(answer);
    }

    public void deleteAnswer(Long answerId) throws Exception {
        try {
            repo.delete(answerId);
        } catch (Exception e) {
            logger.error("Exception occured while trying to delete answer: " + answerId, e);
            throw new Exception("Unable to delete answer.");
        }
    }

}
```

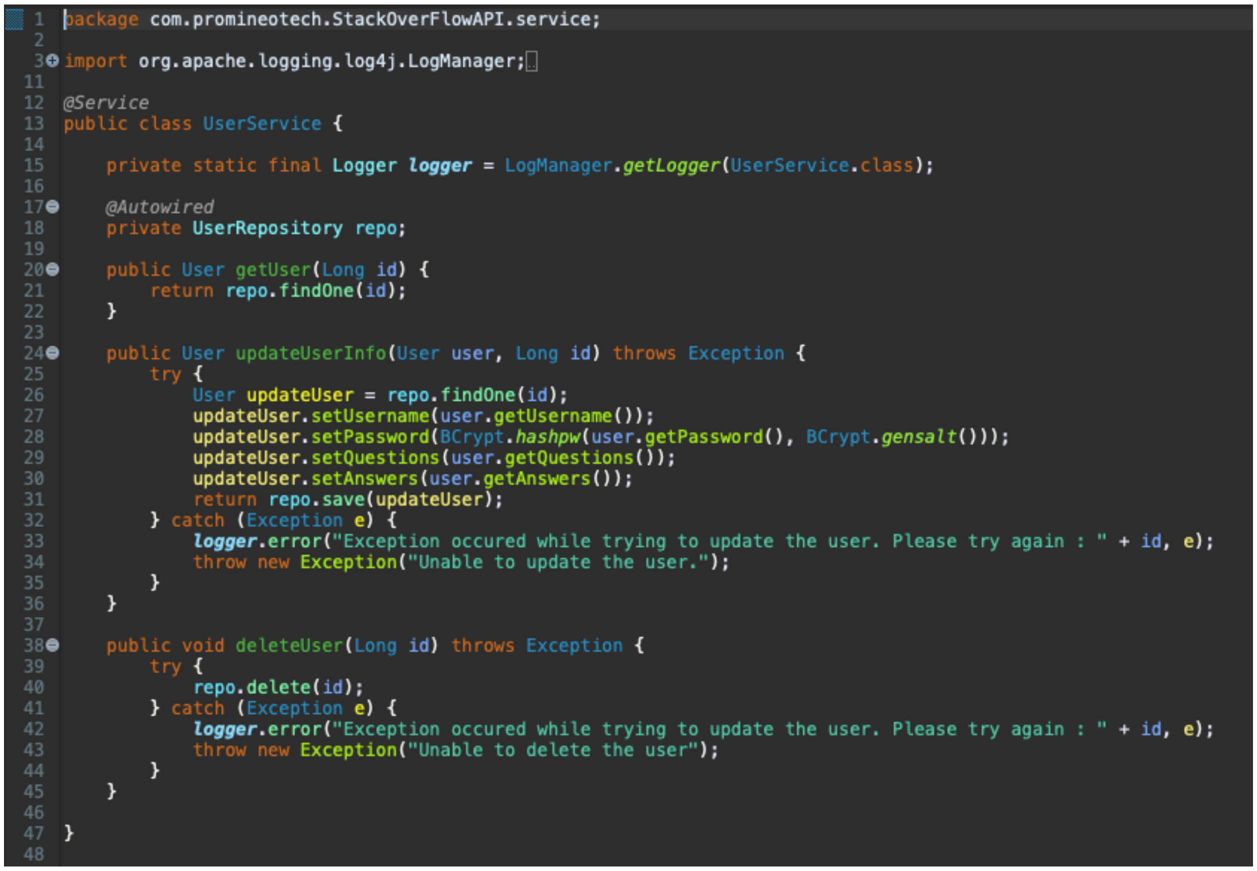


```java
package com.promineotech.StackOverFlowAPI.service;

import javax.naming.AuthenticationException;

@Service
public class AuthService {

    @Autowired
    private UserRepository userRepository;

    public User register(Credentials cred) throws AuthenticationException {
        User user = new User();
        user.setUsername(cred.getUsername());
        user.setPassword(BCrypt.hashpw(cred.getPassword(), BCrypt.gensalt()));
        try {
            userRepository.save(user);
            return user;
        } catch (DataIntegrityViolationException e) {
            throw new AuthenticationException("Username is not valid! Please try again");
        }
    }

    public User login(Credentials cred) throws AuthenticationException {
        User foundUser = userRepository.findbyUsername(cred.getUsername());
        if(BCrypt.checkpw(cred.getPassword(), foundUser.getPassword())) {
            return foundUser;
        }
        throw new AuthenticationException("Incorrect Username or Password. Please try again!");
    }

}
```



```java
package com.promineotech.StackOverFlowAPI.service;

import org.apache.logging.log4j.LogManager;

@Service
public class QuestionService {

    private static final Logger logger = LogManager.getLogger(QuestionService.class);

    @Autowired
    private QuestionRepository repo;

    public Question createQuestion(Question question) {
        return repo.save(question);
    }

    public Question updateQuestion(Question question, Long id) throws Exception {
        try {
            Question oldQuestion = repo.findOne(id);
            oldQuestion.setContent(question.getContent());
            return repo.save(oldQuestion);
        } catch (Exception e) {
            logger.error("Exception occured while trying to update the question: " + id, e);
            throw new Exception("Unable to update your question. Please try again!");
        }
    }

    public void removeQuestion(Long id) throws Exception {
        try {
            repo.delete(id);
        } catch (Exception e) {
            logger.error("Exception occured while trying to delete the question: " + id, e);
            throw new Exception("Unable to delete your question. Please try again!");
        }
    }

}
```

PROMINEO TECH



```java
package com.promineotech.StackOverFlowAPI.service;

import org.apache.logging.log4j.LogManager;

@Service
public class UserService {

    private static final Logger logger = LogManager.getLogger(UserService.class);

    @Autowired
    private UserRepository repo;

    public User getUser(Long id) {
        return repo.findOne(id);
    }

    public User updateUserInfo(User user, Long id) throws Exception {
        try {
            User updateUser = repo.findOne(id);
            updateUser.setUsername(user.getUsername());
            updateUser.setPassword(BCrypt.hashpw(user.getPassword(), BCrypt.gensalt()));
            updateUser.setQuestions(user.getQuestions());
            updateUser.setAnswers(user.getAnswers());
            return repo.save(updateUser);
        } catch (Exception e) {
            logger.error("Exception occured while trying to update the user. Please try again : " + id, e);
            throw new Exception("Unable to update the user.");
        }
    }

    public void deleteUser(Long id) throws Exception {
        try {
            repo.delete(id);
        } catch (Exception e) {
            logger.error("Exception occured while trying to update the user. Please try again : " + id, e);
            throw new Exception("Unable to delete the user");
        }
    }

}
```

URL to GitHub Repository: https://github.com/DesmondYo/StackOverflow-API