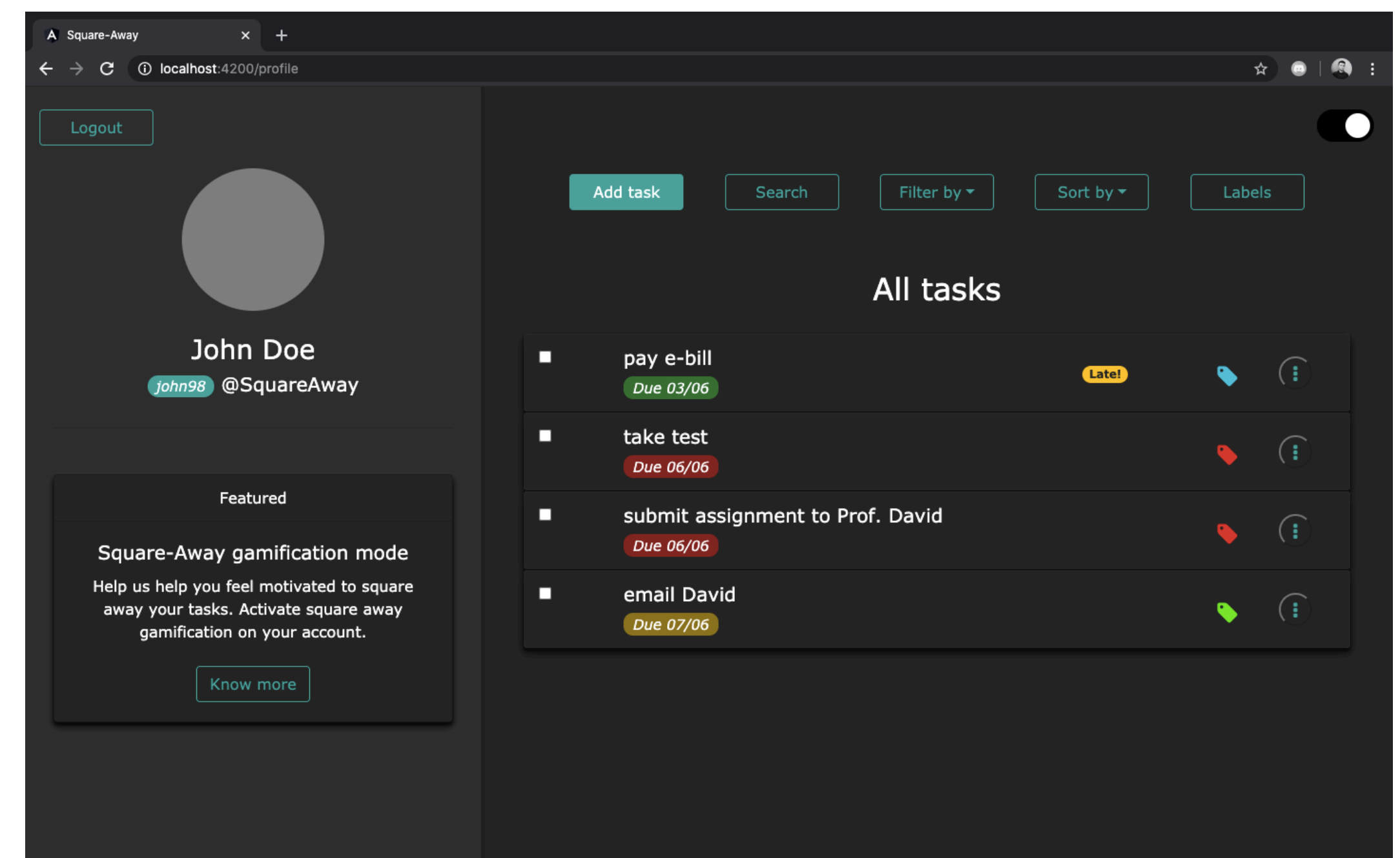
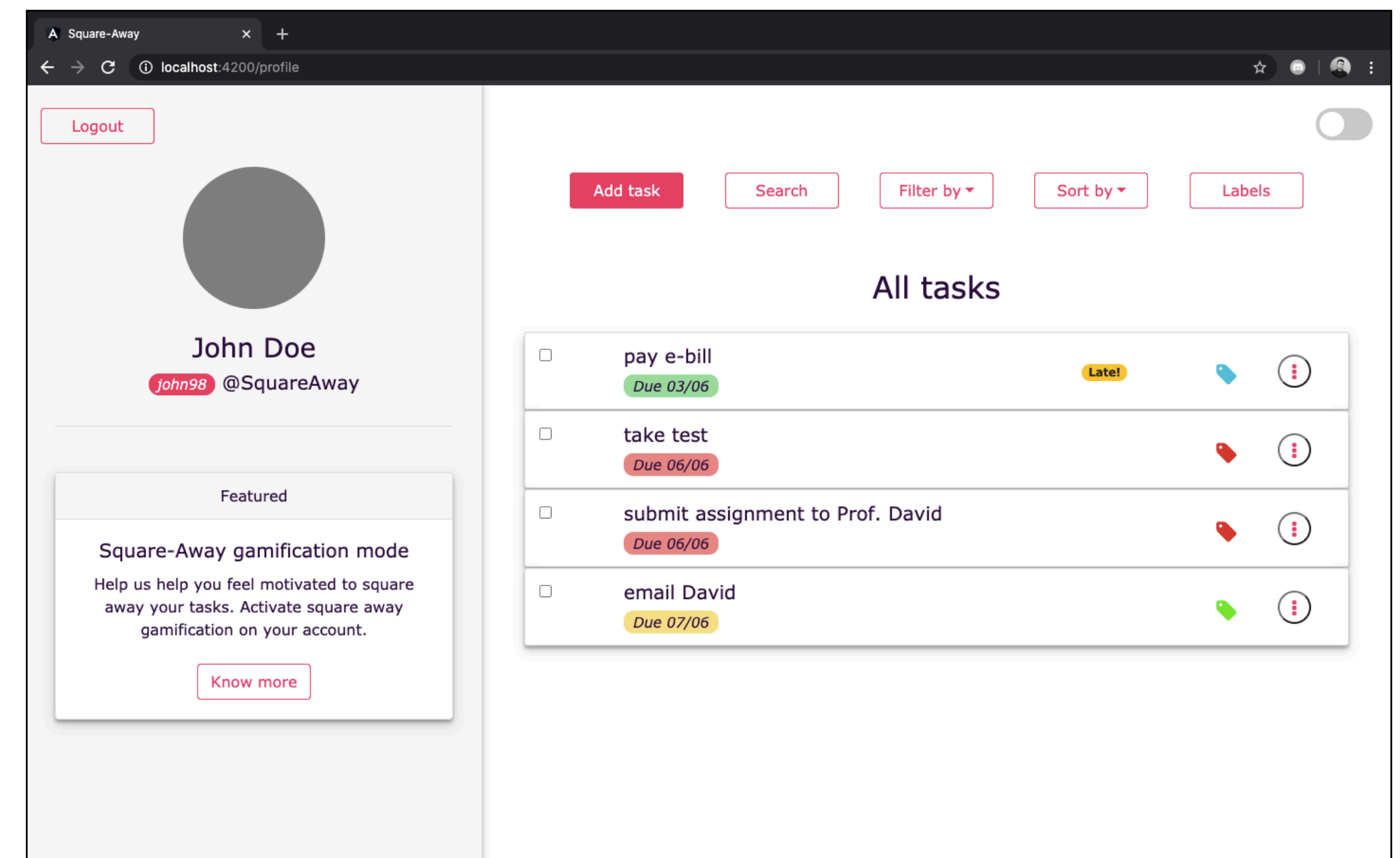


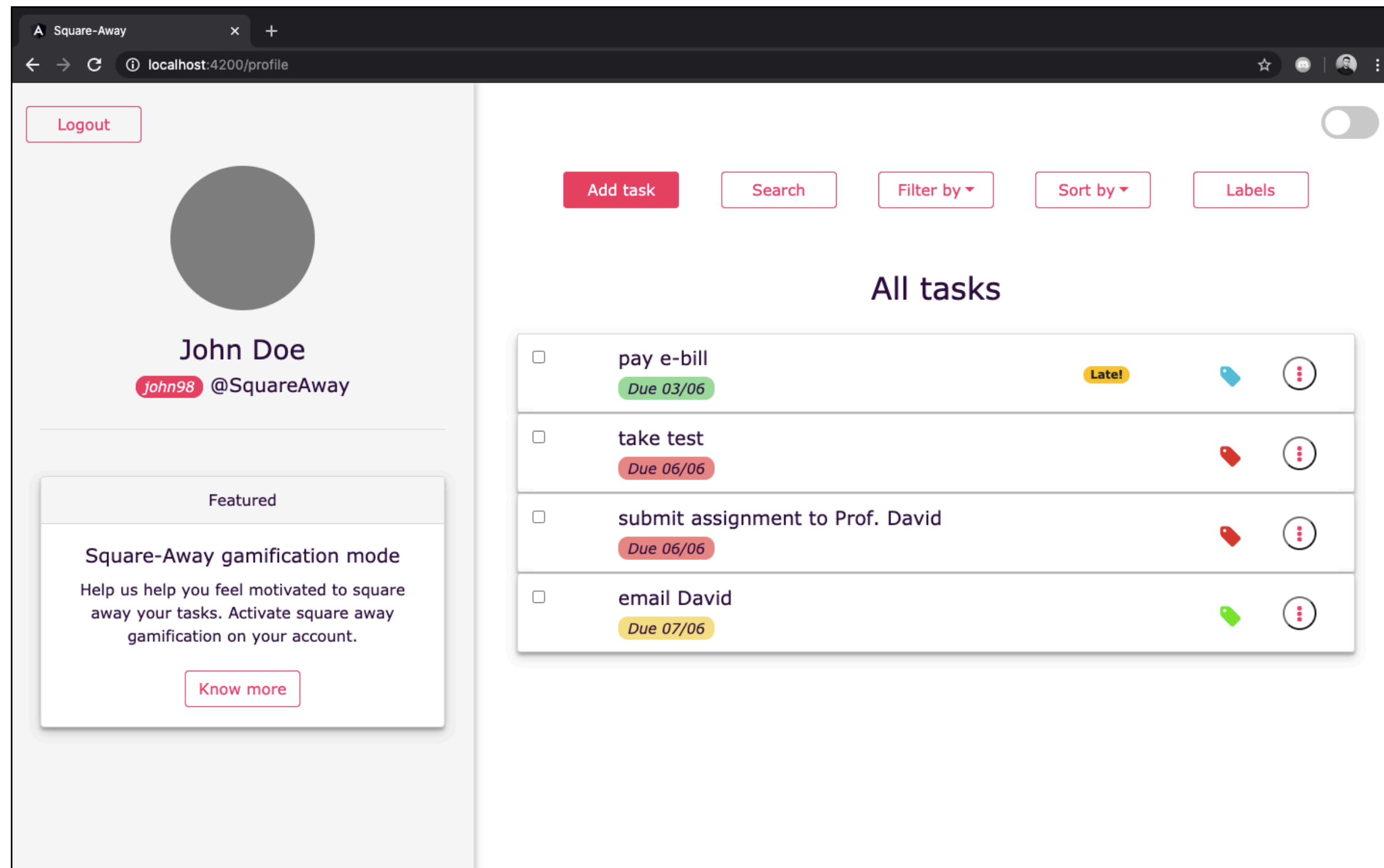
# Square-Away

## StackHack 1.0 Hackathon

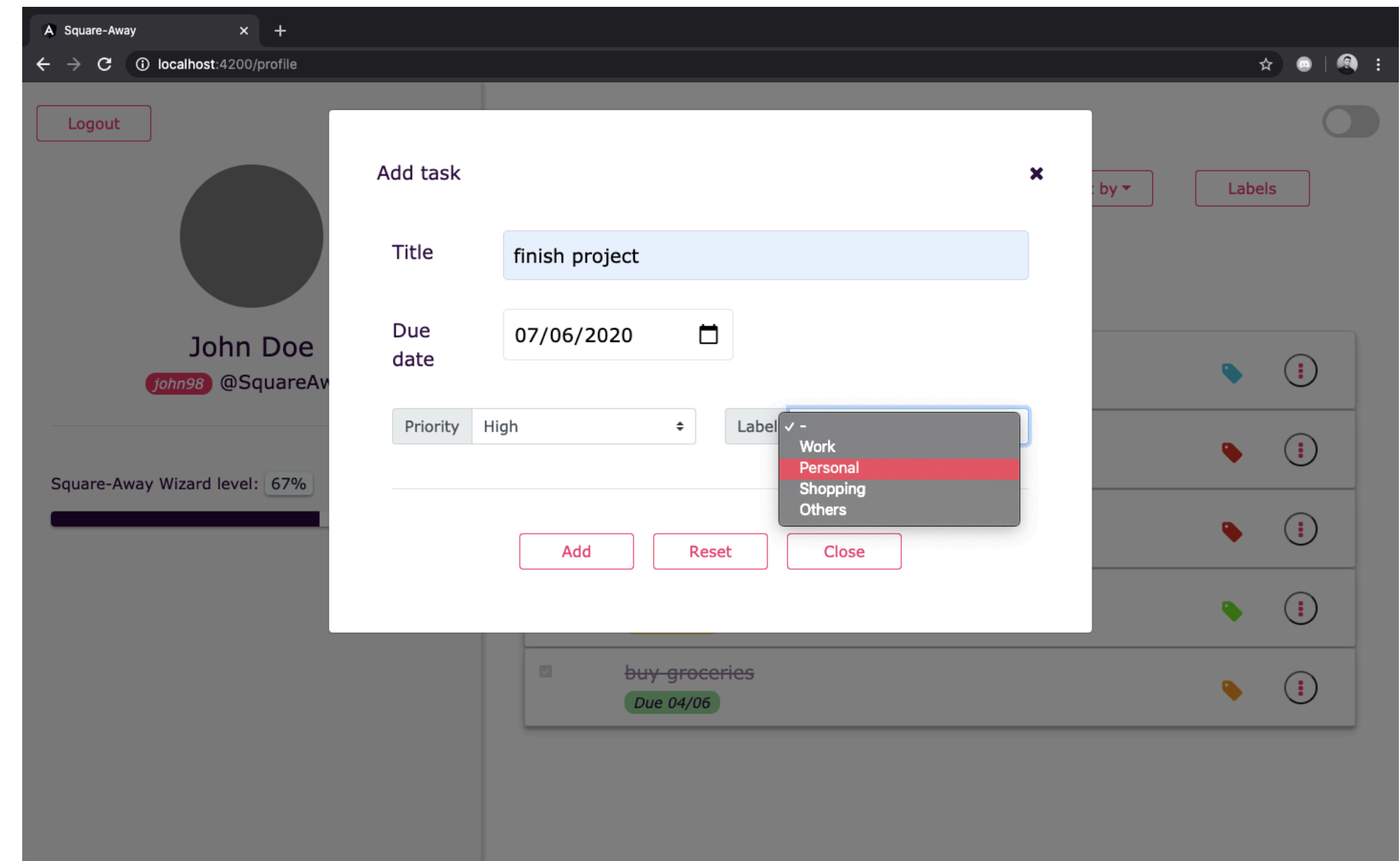
By Rahul Suresh

- **Square-away** is a task manager web-app developed using the **MEAN** stack. (MongoDB, Express, Angular, Node)
- It provides multiple operations for users to efficiently manage their tasks.
- With a **minimal UI**, square-away app provides some features apart from basic task management such as:
  - Filtering, searching, sorting of tasks.
  - UI theme toggle between light and dark theme.
  - Square away gamification

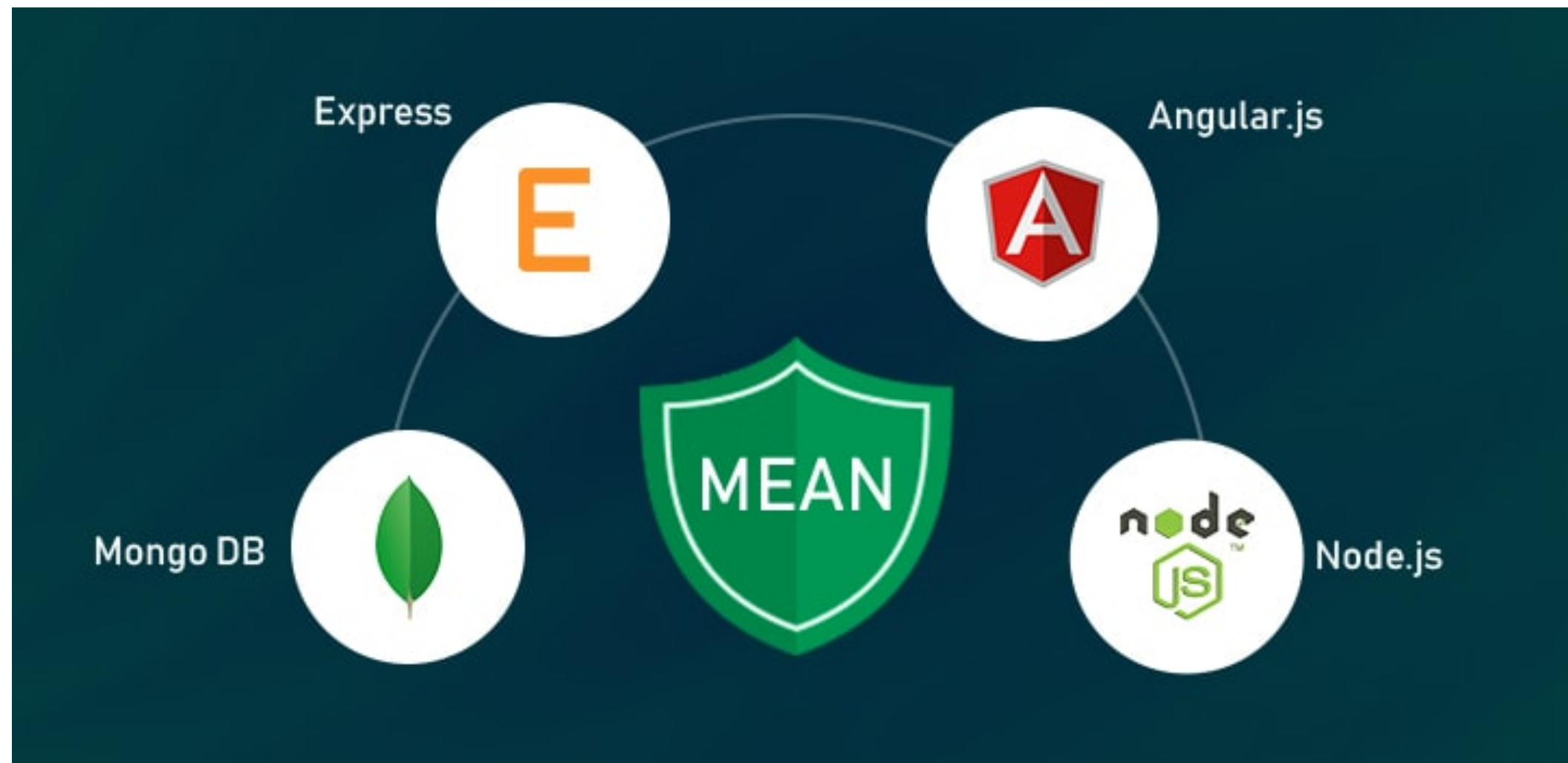




- Each task item can be individually edited, updated or deleted



- New tasks can be added as and when required, along with the due date, priority and label.



- The tech stack used is the MEAN Stack

```
router.get('/:username/readAll', (req, res, next) => {  
  let username = req.params.username;  
  Task.getAllTasks(username, (err, tasks) => {  
    if(err)  
      res.json({success: false, msg: "Failed to retrieve tasks!"});  
    else {  
      console.log("Retrieved tasks: " + tasks.length);  
      res.json({success: true, tasks: tasks});  
    }  
  });  
});
```

```
getAllTasks(username:string):Observable<any> {  
  return this.http.get(  
    this.prefix + username + this.getAllTasksURL  
  ).map(res => res.json());  
}
```

- Data is shared between the Frontend and Backend in JSON format, over REST APIs.



```
Rahuls98s-MacBook-Air:components rahuls98$ ls
add-task      label-modal   profile        register       task-item
edit-task     login         profile-body   sag-modal      view-task
home          navbar        profile-sidenav search-bar
```

- Since the app is built using Angular, it is very modular making use of multiple components.

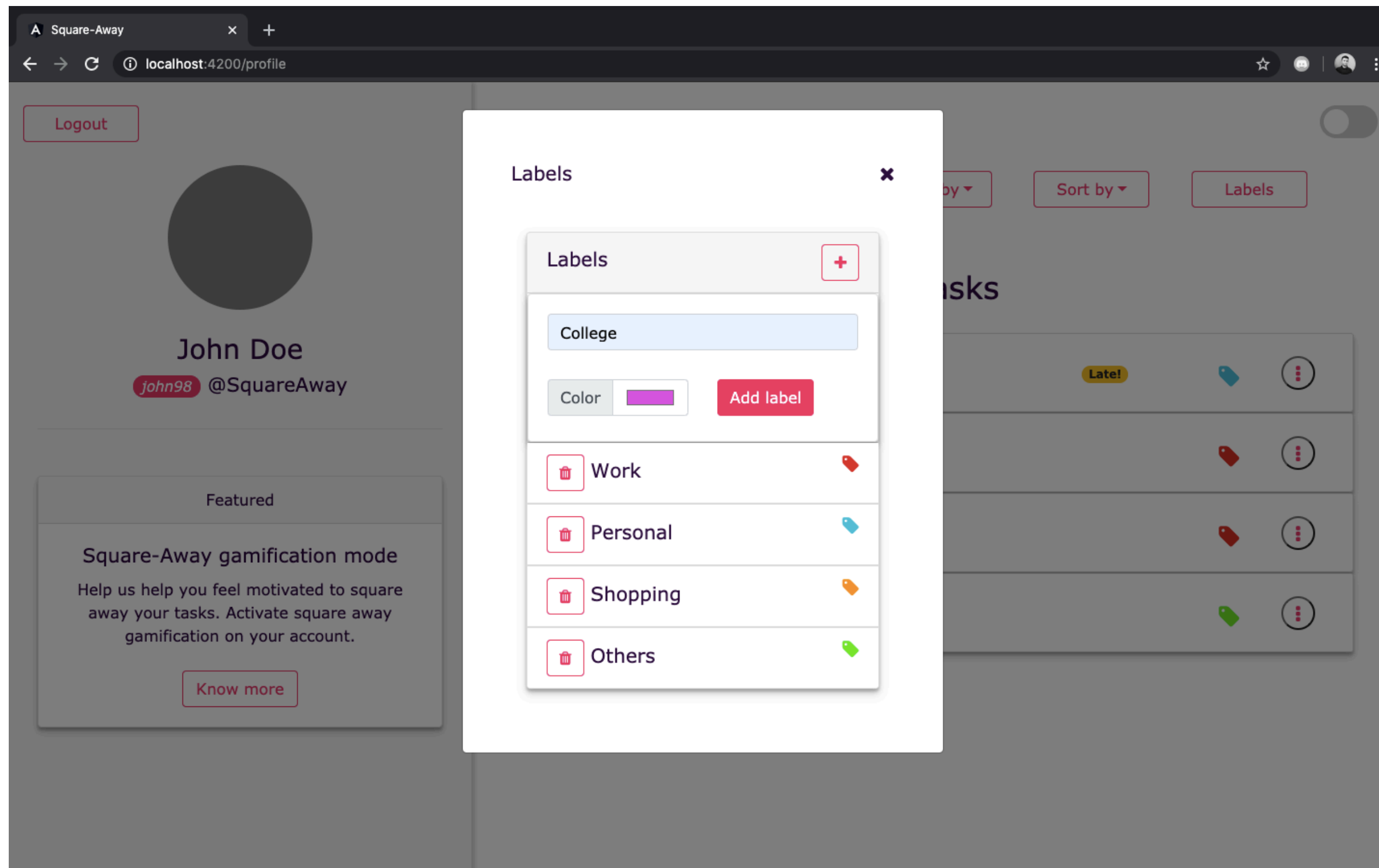
```
//TASK SCHEMA
const TaskSchema = mongoose.Schema({
  username: { type: String, require: true },
  title: { type: String, require: true },
  dueDate: { type: Date, min: getPreviousDate(), require: true },
  priority: { type: String, require: true },
  label: { type: String, require: true },
  status: { type: String, require: true },
  isDone: { type: Boolean, require: true },
  gamification: {
    checked: { type: Date, require: true },
    score : { type: Number, require: true }
  }
});

const Task = module.exports = mongoose.model('Task', TaskSchema);
```

```
//USER SCHEMA
const UserSchema = mongoose.Schema({
  name: { type: String, require: true },
  email: { type: String, require: true },
  username: { type: String, require: true },
  password: { type: String, require: true },
  labels: { type: Array, require: true },
  sag: { type: Boolean, require: true },
  gamification: {
    activeOn: { type: Array, require: true },
    score: { type: Number, require: true },
    n : { type: Number, require: true }
  }
});

const User = module.exports = mongoose.model('User', UserSchema);
```

- User and Task schemas are used for the backend database services.

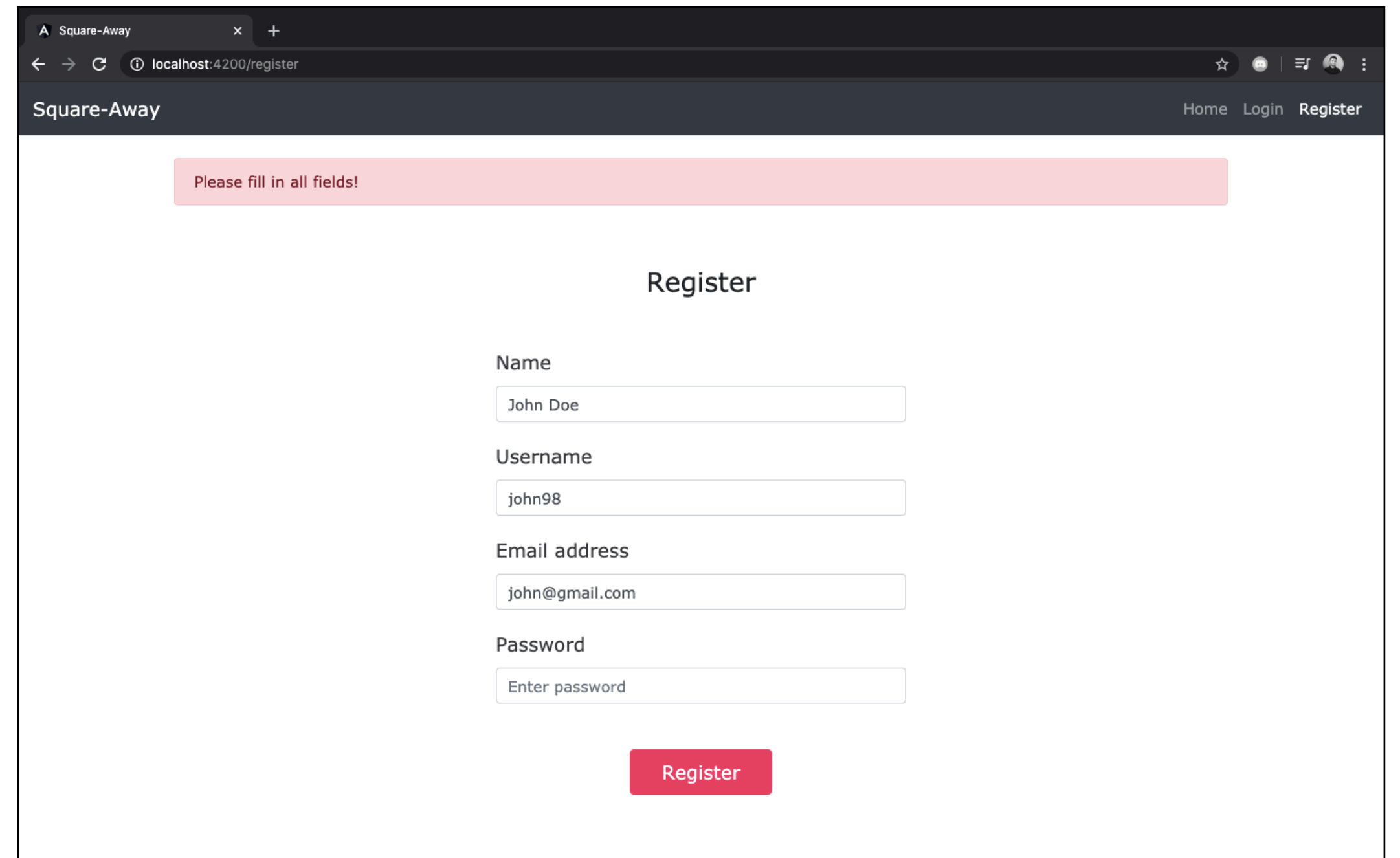


- **New labels** can be added/deleted as per user requirement, and they can be applied to the corresponding tasks to be categorized.

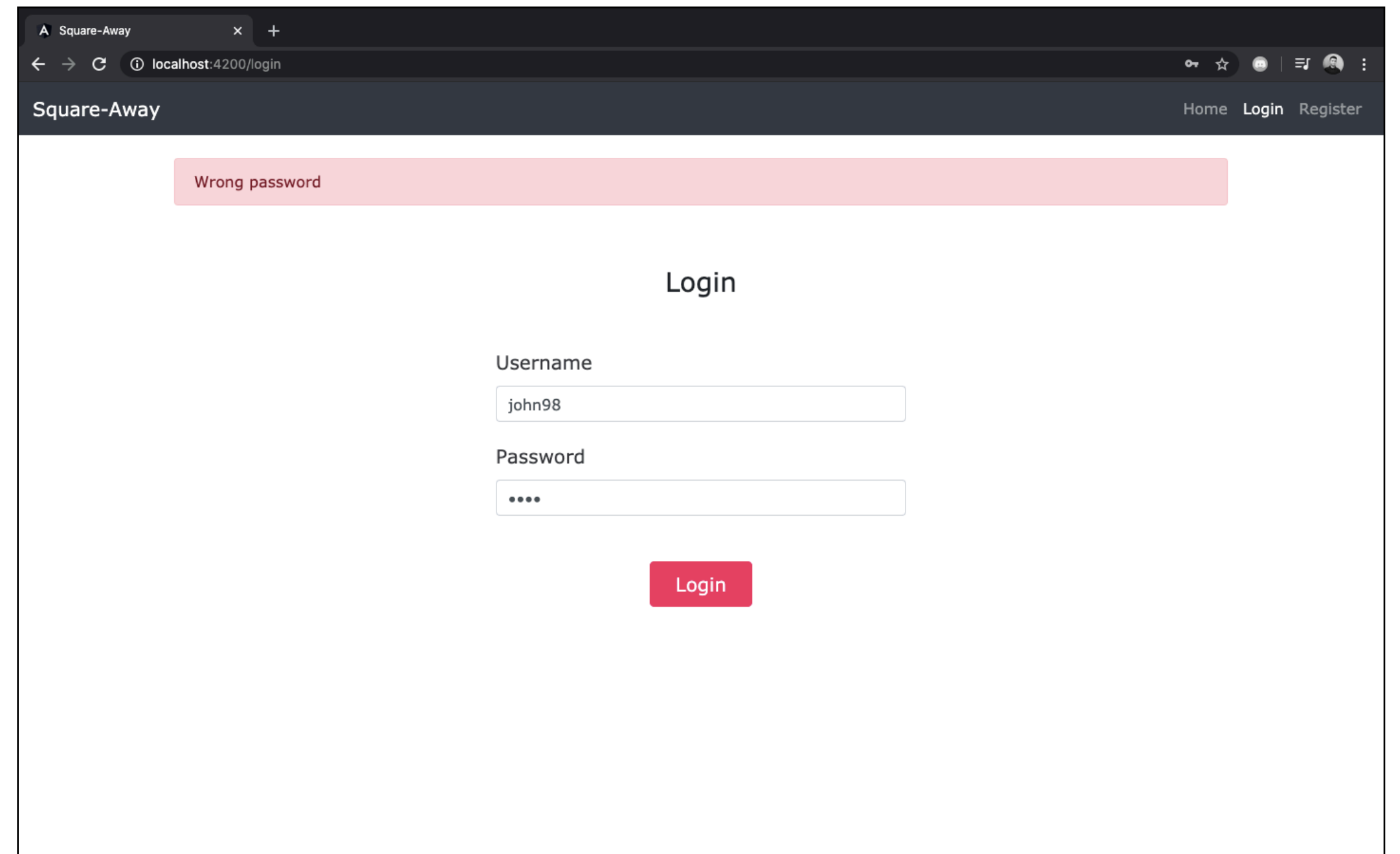
```
"labels" : [
  {
    "name" : "Work",
    "color" : "#d42121"
  },
  {
    "name" : "Personal",
    "color" : "#1dbcd7"
  },
  {
    "name" : "Shopping",
    "color" : "#ff9500"
  },
  {
    "name" : "Others",
    "color" : "#18e225"
  }
],
"name" : "John Doe",
"email" : "john@gmail.com",
"username" : "john98",
```

- These account specific labels are maintained in the database, and are retrieved by the frontend as and when required.

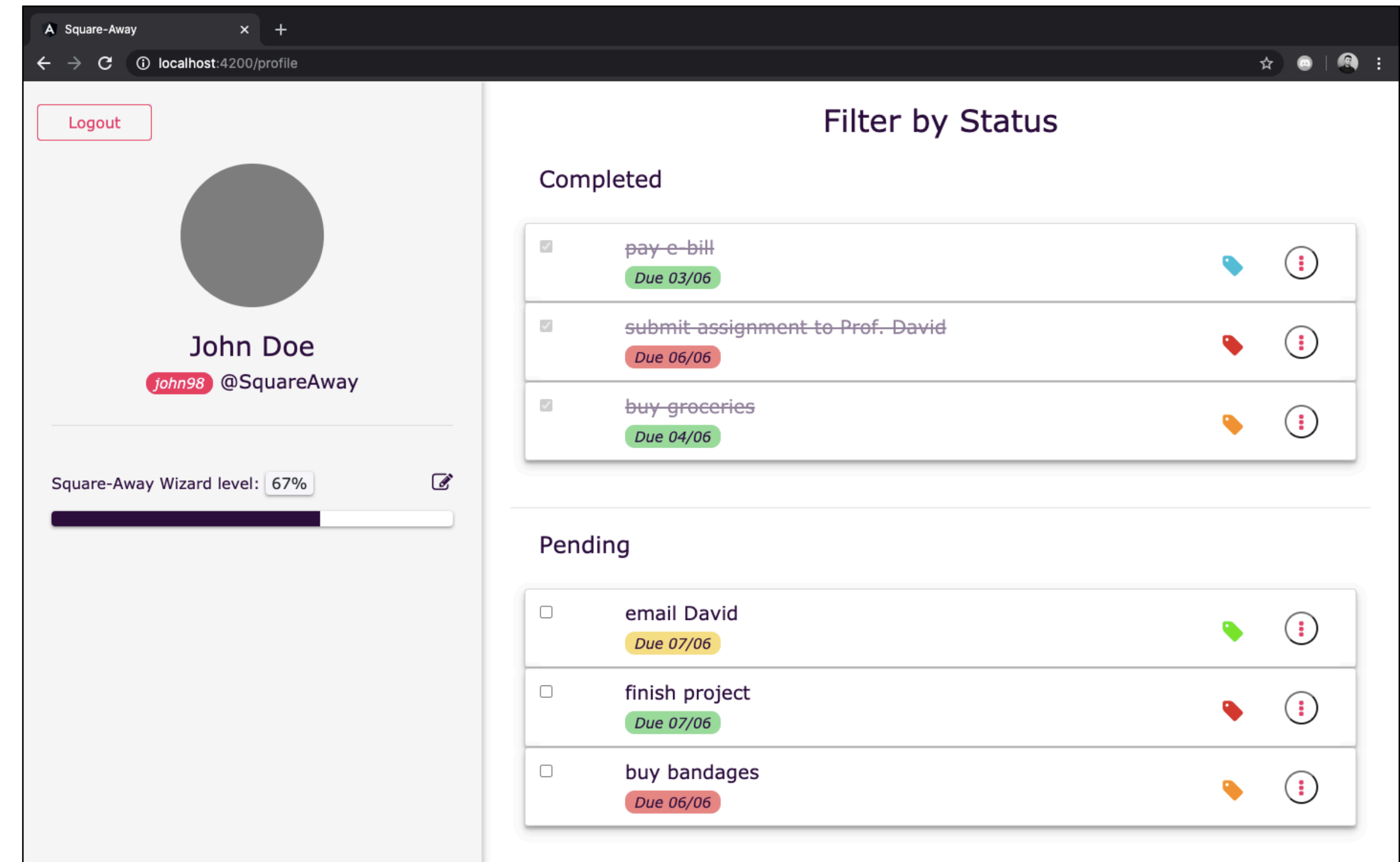
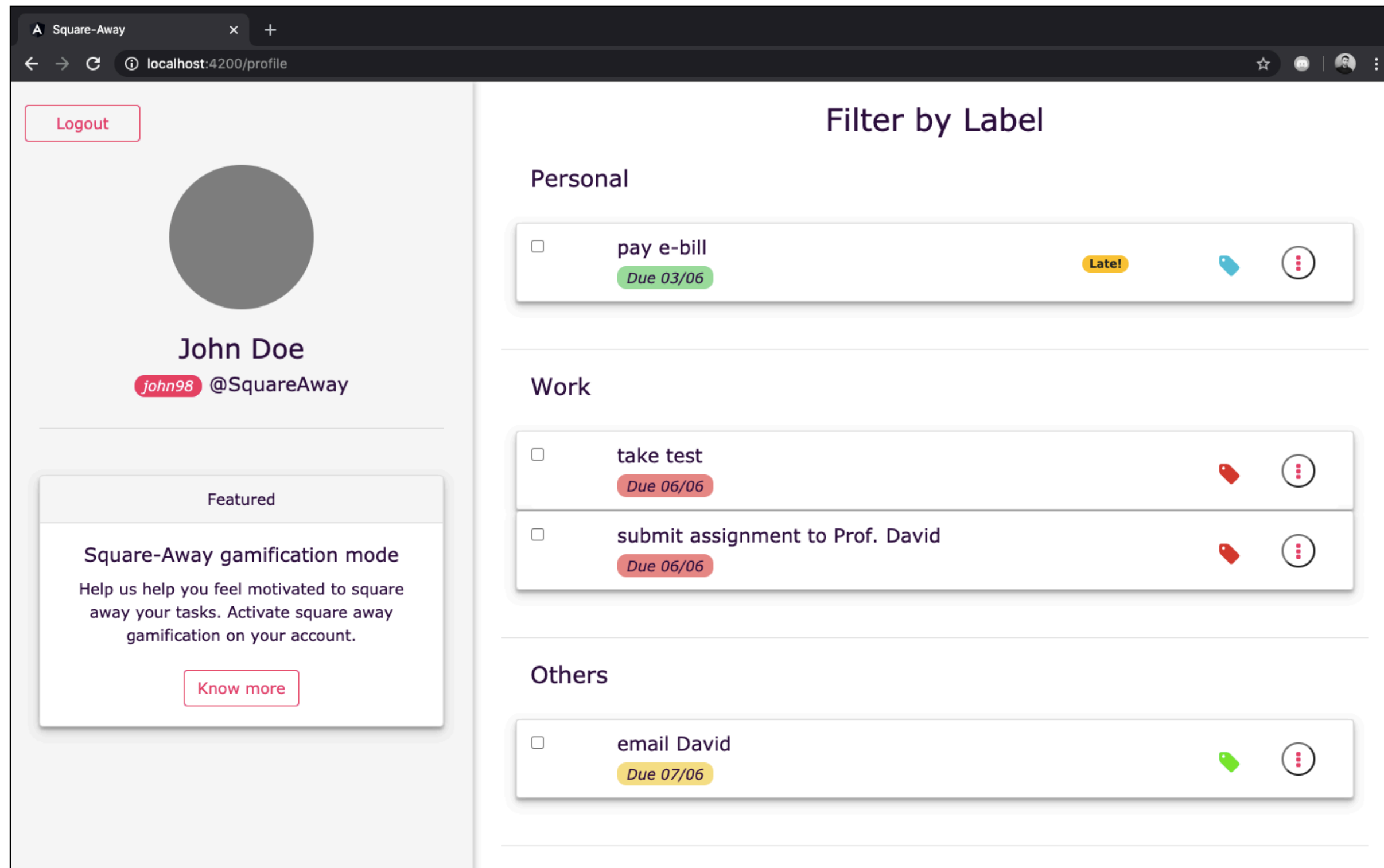
- **Signup and Login/Logout** functionality is implemented in the Square-Away app, with user-auth schema in the database.
- Passwords are hashed using **bcryptJS** before storing in the database.
- **Passport** and JSON web token (**JWT**) are used to authenticate requests to specific url's.
- **Form validation** is also done on both the Login and Registration forms.



A screenshot of a web browser showing the 'Square-Away' application's registration page. The browser's address bar shows 'localhost:4200/register'. The page has a dark header with 'Square-Away' and navigation links 'Home', 'Login', and 'Register'. A pink error message at the top says 'Please fill in all fields!'. The 'Register' form includes fields for 'Name' (filled with 'John Doe'), 'Username' (filled with 'john98'), 'Email address' (filled with 'john@gmail.com'), and 'Password' (placeholder 'Enter password'). A red 'Register' button is at the bottom.

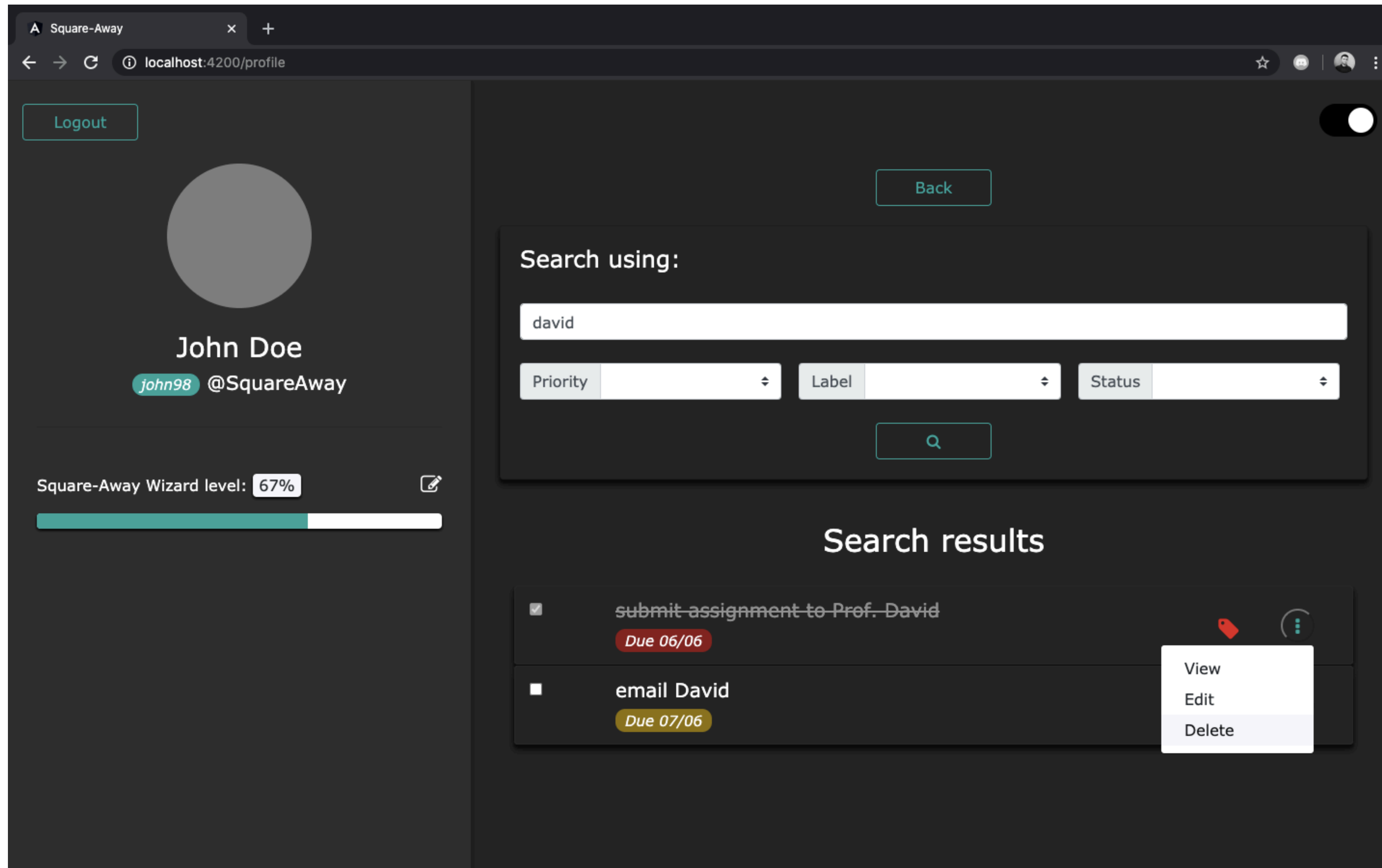


A screenshot of a web browser showing the 'Square-Away' application's login page. The browser's address bar shows 'localhost:4200/login'. The page has a dark header with 'Square-Away' and navigation links 'Home', 'Login', and 'Register'. A pink error message at the top says 'Wrong password'. The 'Login' form includes fields for 'Username' (filled with 'john98') and 'Password' (masked with '\*\*\*\*'). A red 'Login' button is at the bottom.



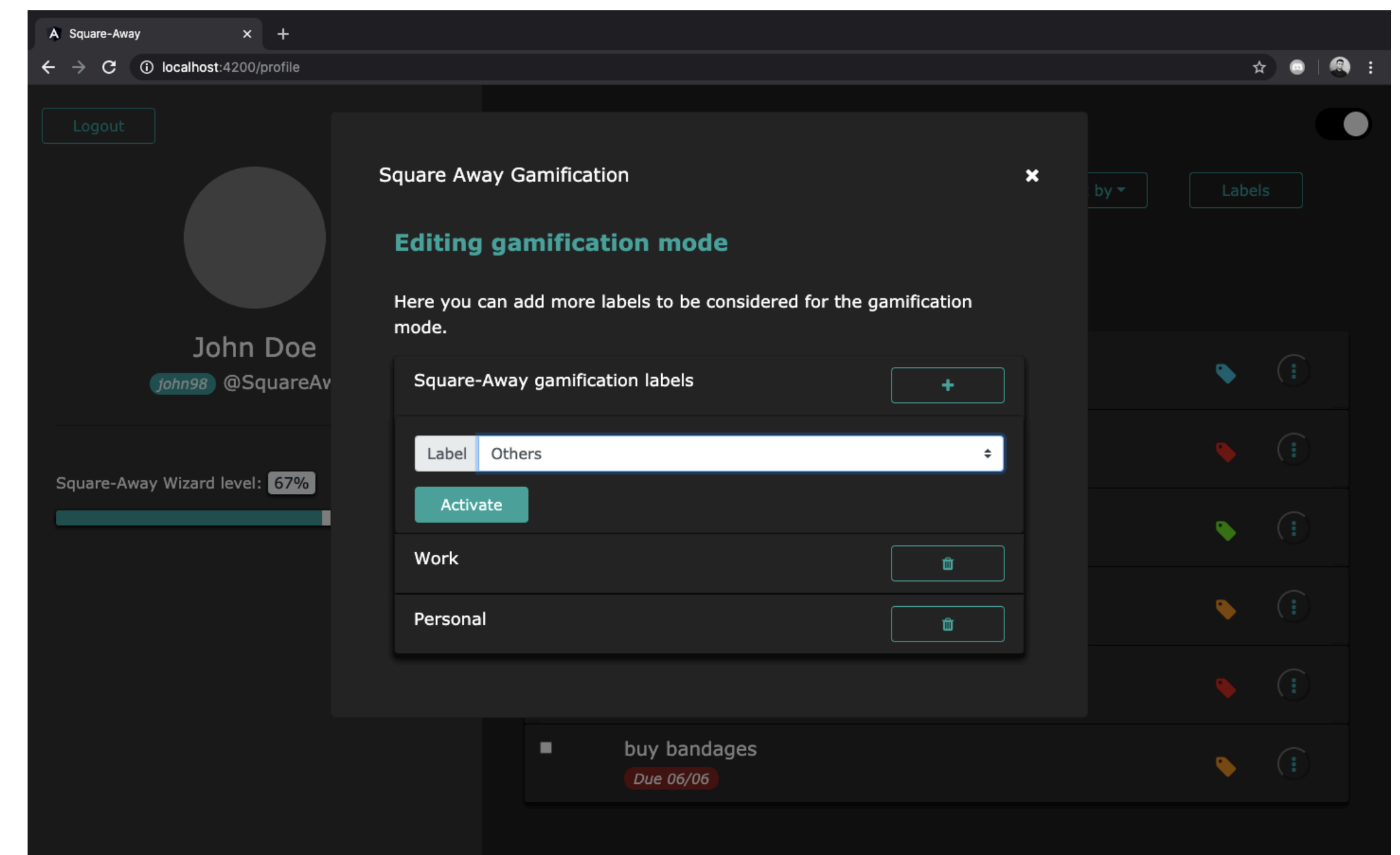
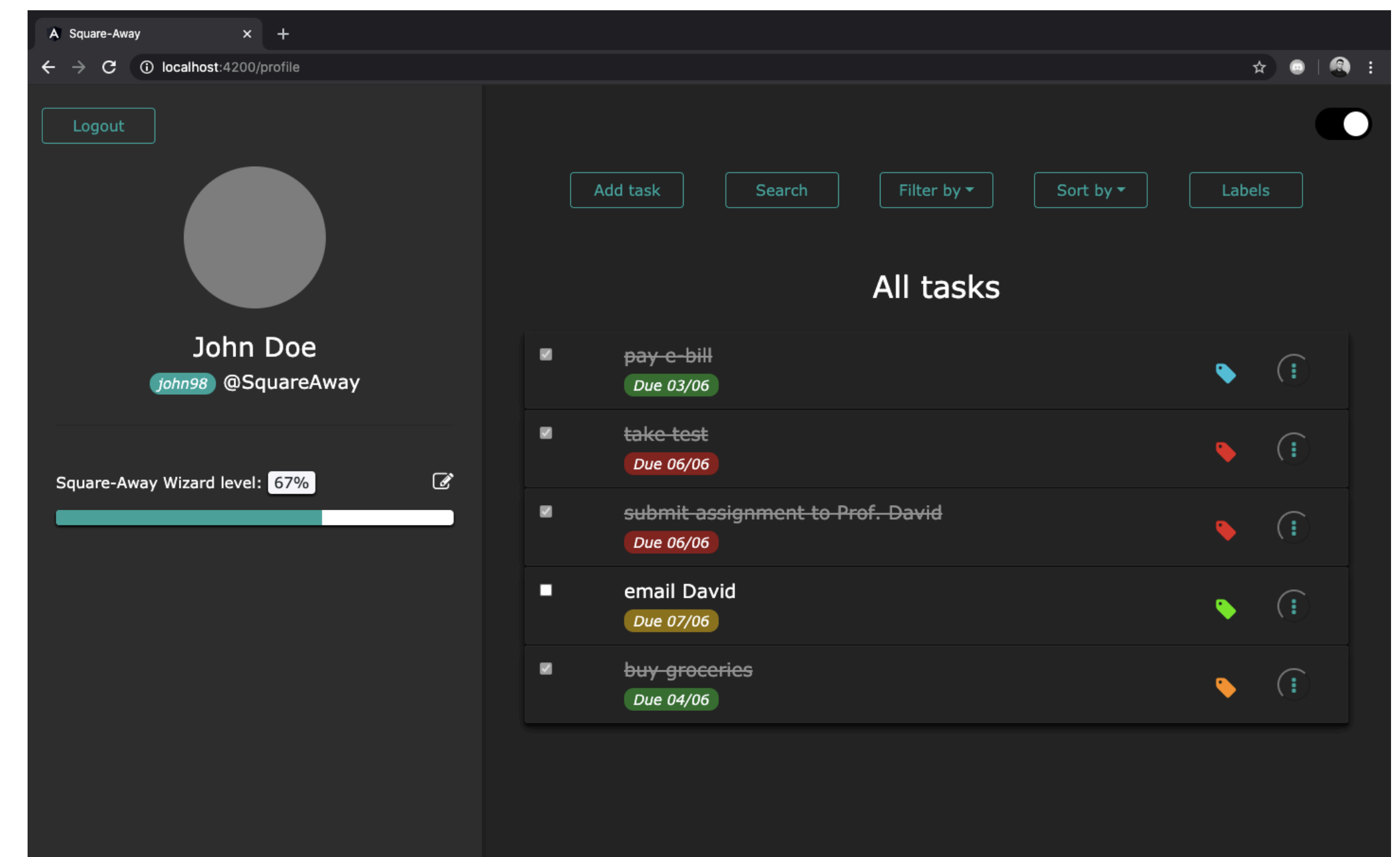
- Tasks can be **filtered** using label, priority or status of completion on the click of a button.
- Apart from filtering, tasks can also be **sorted** using the dueDate.





- Tasks from the task-list can be **searched** for using **specific** keywords / title, priority, label, status, or any **combination** of all the 4 parameters

- **Square Away Gamification mode** is a feature that helps users feel motivated to complete tasks. Once the feature is enabled, the process of task completion is gamified by the inclusion of a **point scoring strategy**.
- For every task completed, a score in the range  $[0,2]$  is assigned to the user based on **early/late task completion**, adding/reducing the overall score.
- Users can select which categories of tasks to apply gamification to, eg: Projects, Office Work, etc.
- The gamification score can act as a requirement for specific account features / UI elements in the future.



**Thank You!**