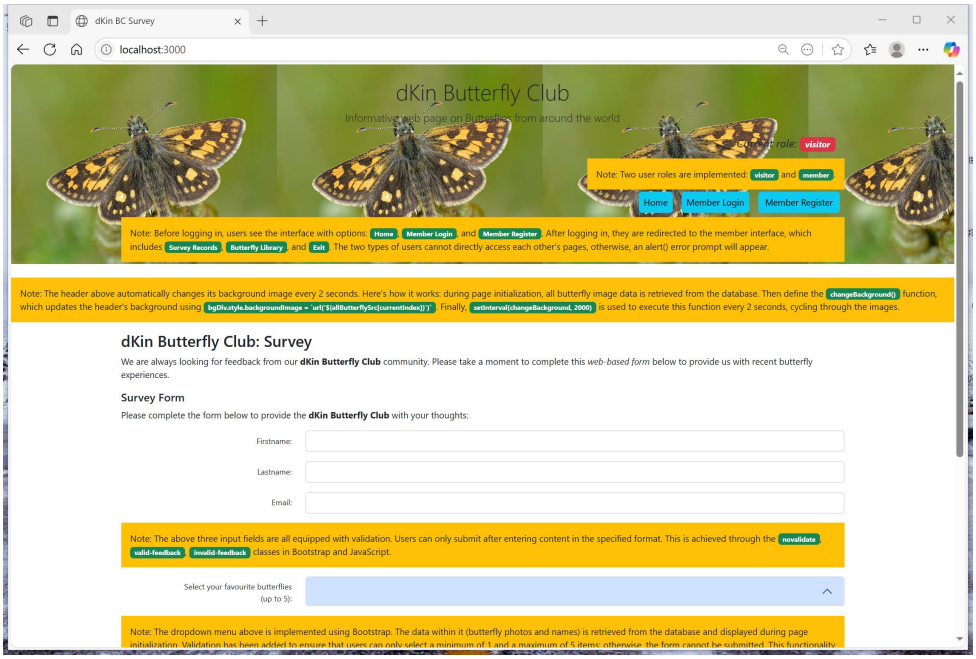
Link to source code repository (for localhost deployment)

https://github.com/6839/MyWebProject.git

Branch: **main**

Code Feature: use local MySQL database on my computer, can't be hosted externally

Demo:



Link to source code repository (for external host deployment)

https://github.com/6839/MyWebProject.git

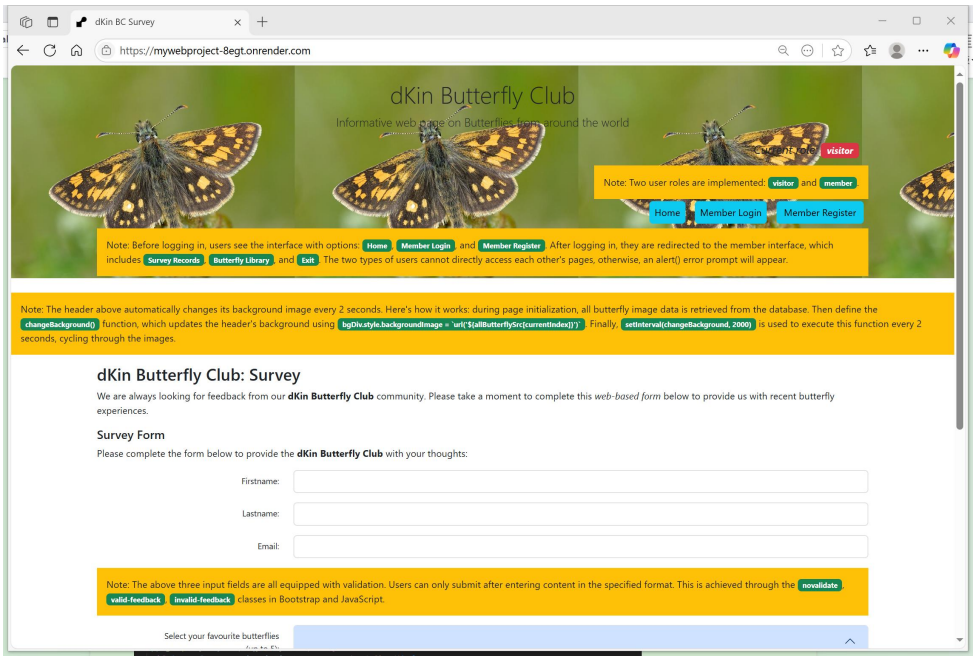
Branch: **external**

Code Feature: use built-in sqlite3 database, can be hosted externally

Test link: https://mywebproject-8egt.onrender.com

Restriction: It automatically goes into sleep mode after 15 minutes of no requests, and is suitable for non-continuously high traffic applications.

Demo:



Link to project video

https://deakin.au.panopto.com/Panopto/Pages/Viewer.aspx?id=3a7c818b-3021-4533-8e7d-b2de00aae868

Brief introduction and overview of the topic/features

The website has made some modifications and functional extensions based on the dKin Butterfly Club survey, allowing users to check their favorite butterflies. In addition, it supports member registration and login. After logging in, members can view all submitted surveys and information about all butterflies. The detailed features are as follows:

- Header background image switch automatically

index.js

```
// visitor get index page
app.get('/', async (req, res, next) => {
  if (typeof req.session.isFirstAccess === 'undefined') {
    req.session.isFirstAccess = true;
  }

  if (req.session.isFirstAccess) {
    req.session.allimgSrc = await runMysqlQuery('select imgSrc from butterfly');
    req.session.allButterflySrc = req.session.allimgSrc.map(row => row.imgSrc);
    req.session.role = 0;
    req.session.isFirstAccess = false;
  }

  if (req.session.role == 0) {
    req.session.visit_verificationCode = '';
    req.session.visit_verificationCode = generateVerificationCode(req.session.visit_verificationCode);
    const total_butterfly = await runMysqlQuery('select * from butterfly');
    res.render('index', {
      title: 'dKin BC Survey',
      butterflyList: total_butterfly,
      verificationCode: req.session.visit_verificationCode,
      message_prompt: res.locals.message_prompt,
      allButterflySrc: req.session.allButterflySrc,
      displayNote: true,
      role: req.session.role
    });
```

header.ejs

```
<script>
  const allButterflySrc = '<%= allButterflySrc %>'.split(",");
  let currentIndex = 0;
  const bgDiv = document.getElementById('bgDiv');

  function changeBackground() {
    bgDiv.style.backgroundImage = `url('${allButterflySrc[currentIndex]}')`;
    currentIndex = (currentIndex + 1) % allButterflySrc.length;
  }

  setInterval(changeBackground, 2000);

</script>
```

- Create butterfly card list for users to choose when filling the survey

index.js

```
if (req.session.role == 0) {
  req.session.visit_verificationCode = '';
  req.session.visit_verificationCode = generateVerificationCode(req.session.visit_verificationCode);
  const total_butterfly = await runMysqlQuery('select * from butterfly');
  res.render('index', {
    title: 'dKin BC Survey',
    butterflyList: total_butterfly,
    verificationCode: req.session.visit_verificationCode,
    message_prompt: res.locals.message_prompt,
    allButterflySrc: req.session.allButterflySrc,
    displayNote: true,
    role: req.session.role
  });
```

index.ejs

```
<div class="accordion-item">
    <h2 class="accordion-header" id="headingOne">
        <button class="accordion-button" type="button" data-bs-toggle="collapse"
            data-bs-target="#collapseOne">
        </button>
    </h2>
    <div id="collapseOne" class="accordion-collapse collapse"
        data-bs-parent="#butterflyAccordion">
        <div class="accordion-body d-flex flex-wrap">
            <% for(let i=0; i < butterflyList.length; i++){ %>
                <div class="col-md-3 mb-2">
                    <img src="<%= butterflyList[i].imgSrc %>"
                        alt="<%= butterflyList[i].commonName %>"
                        class="img-fluid img-thumbnail card-img-top">
                    <div class="text-center">
                        <input type="checkbox" id="<%= butterflyList[i].id %>"
                            name="<%= butterflyList[i].id %>"
                            class="form-check-input">
                        <label for="<%= butterflyList[i].commonName %>"
                            class="text-secondary form-check-label">
                            <%= butterflyList[i].commonName %>
                        </label>
                    </div>
                </div>
                <% } %>
        </div>
    </div>
```

index.js

```
app.post('/surveySubmit', async (req, res, next) => {
  // get data from user's input
  let inputFirstname = req.body.firstname;
  let inputLastname = req.body.lastname;
  let inputEmail = req.body.email;
  let inputFavouriteButterfly = [];
  let inputComments = req.body.comments;
  let inputValidationCode = req.body.validationCode;

  for (const key in req.body) {
    if (req.body[key] == 'on') {
      inputFavouriteButterfly.push(key);
    }
  }
```

- The previous image-based verification code has been changed to the internal code.

index.js

```
function generateVerificationCode(code) {
  for (let i = 0; i < 6; i++) {
    code += Math.floor(Math.random() * 10);
  }
  return code;
}
```

```
if (req.session.role == 0) {
  req.session.visit_verificationCode = '';
  req.session.visit_verificationCode = generateVerificationCode(req.session.visit_verificationCode);
  const total_butterfly = await runMysqlQuery('select * from butterfly');
  res.render('index', {
    title: 'dKin BC Survey',
    butterflyList: total_butterfly,
    verificationCode: req.session.visit_verificationCode,
    message_prompt: res.locals.message_prompt,
    allButterflySrc: req.session.allButterflySrc,
    displayNote: true,
    role: req.session.role
  });
} else {
```

index.ejs

```
<div class="col-md-auto">
    <label class="fst-italic bg-info fw-bold fs-4 p-1 mx-2"><%- verificationCode %></label>
</div>
```

- Add user registration and login function. After logging in, users can view all the survey records submitted, and then calculate the top 3 favorite butterflies and display them. Logged in users can also see the list of all butterflies (including their Common Name, Scientific Name and the cumulative number of user favorites).

- Add a number of security checks, such as whether the two passwords are the same when registering, checking whether the user has been registered when logging in, whether the password is the same, visitor can not access the member page error prompts, whether there is an authorization code when the user is registered, the same user repeated registration error prompts and so on.

index.js

```
index.js:87:      message_prompt: res.locals.message_prompt,
index.js:91:    req.session.message_prompt = 'wrong_access';
index.js:104:     message_prompt: res.locals.message_prompt,
index.js:108:   req.session.message_prompt = 'wrong_access';
index.js:134:   message_prompt: res.locals.message_prompt,
index.js:145:     message_prompt: res.locals.message_prompt,
index.js:149:   req.session.message_prompt = 'wrong_access';
index.js:160:     message_prompt: res.locals.message_prompt,
index.js:164:   req.session.message_prompt = 'wrong_access';
index.js:178:     message_prompt: res.locals.message_prompt,
index.js:182:   req.session.message_prompt = 'wrong_access';
index.js:193:     message_prompt: res.locals.message_prompt,
index.js:197:   req.session.message_prompt = 'already_login_to_login';
index.js:208:     message_prompt: res.locals.message_prompt,
index.js:212:   req.session.message_prompt = 'already_login_to_register';
index.js:225:     message_prompt: res.locals.message_prompt,
index.js:229:   req.session.message_prompt = 'visitor_fail_access_admin_content';
index.js:242:     message_prompt: res.locals.message_prompt,
index.js:246:   req.session.message_prompt = 'visitor_fail_access_admin_content';
index.js:259:     message_prompt: res.locals.message_prompt,
index.js:263:   req.session.message_prompt = 'visitor_fail_access_admin_content';
index.js:276:     message_prompt: res.locals.message_prompt,
index.js:280:   req.session.message_prompt = 'visitor_fail_access_admin_content';
index.js:289:   req.session.message_prompt = 'visitor_fail_access_admin_content';
index.js:307:  req.session.message_prompt = 'feedback_submit_finish'
index.js:318:   req.session.message_prompt = 'user_not_exist';
index.js:322:     req.session.message_prompt = 'login_wrong_password';
index.js:333:       message_prompt: res.locals.message_prompt,
index.js:353:   req.session.message_prompt = 'user_existed';
index.js:367:   req.session.message_prompt = 'register_finish'
```

footer.ejs

```
if('<%= message_prompt %>' == 'invalid'){
    alert("Sorry, there seems to be a problem with the validationCode you supplied.")
}else if ('<%= message_prompt %>' == 'wrong_access') {
    alert("Sorry, as a member, you are not allowed to view this page.")
} else if ('<%= message_prompt %>' == 'already_login_to_login') {
    alert("Sorry, you have already logged in as a member.")
} else if ('<%= message_prompt %>' == 'already_login_to_register') {
    alert("Sorry, you have already logged in as a member, please exit first and then register.")
} else if ('<%= message_prompt %>' == 'visitor_fail_access_member_content') {
    alert("Sorry, as a visitor, you are not allowed to view this page.")
} else if ('<%= message_prompt %>' == 'member_not_exist') {
    alert("Sorry, the member does not exist, please register first.")
} else if ('<%= message_prompt %>' == 'login_wrong_password') {
    alert("Sorry, the password you provided is wrong, please check again.")
} else if ('<%= message_prompt %>' == 'wrong_grantcode') {
    alert("Sorry, the grantcode you provided is not matched, please check and register again.")
} else if ('<%= message_prompt %>' == 'register_existed') {
    alert("Sorry, the user has already existed, please choose another username or login.")
} else if ('<%= message_prompt %>' == 'register_finish') {
    alert("Congratulation!, you has finished your registration, please login.")
}
```

- Add global user access control on the navigation bar, the page displayed before login is the visitor's page, and the page after login is the other pages.

- Different clients can access the site independently, for example, Google Chrome is used to keep the user stay login and check real-time statistics, while edge browser is used for visitors to submit surveys and register members.

- Use the badge and Pagination components in bootstrap to enhance the display effect

- The data in the website are stored and processed using an external MySQL database.

- Add an API interface to view all the registered user information showed by json format

Web page: see web browser(http://localhost:3000/membersapi)

index.js

```
app.get('/membersapi', async (req, res, next) => {
  // Retrieve data from table User on the server
  // and return it in a JSON response
  req.session.members_rows = await runMysqlQuery('select * from member');
  res.json(req.session.members_rows)
});
```