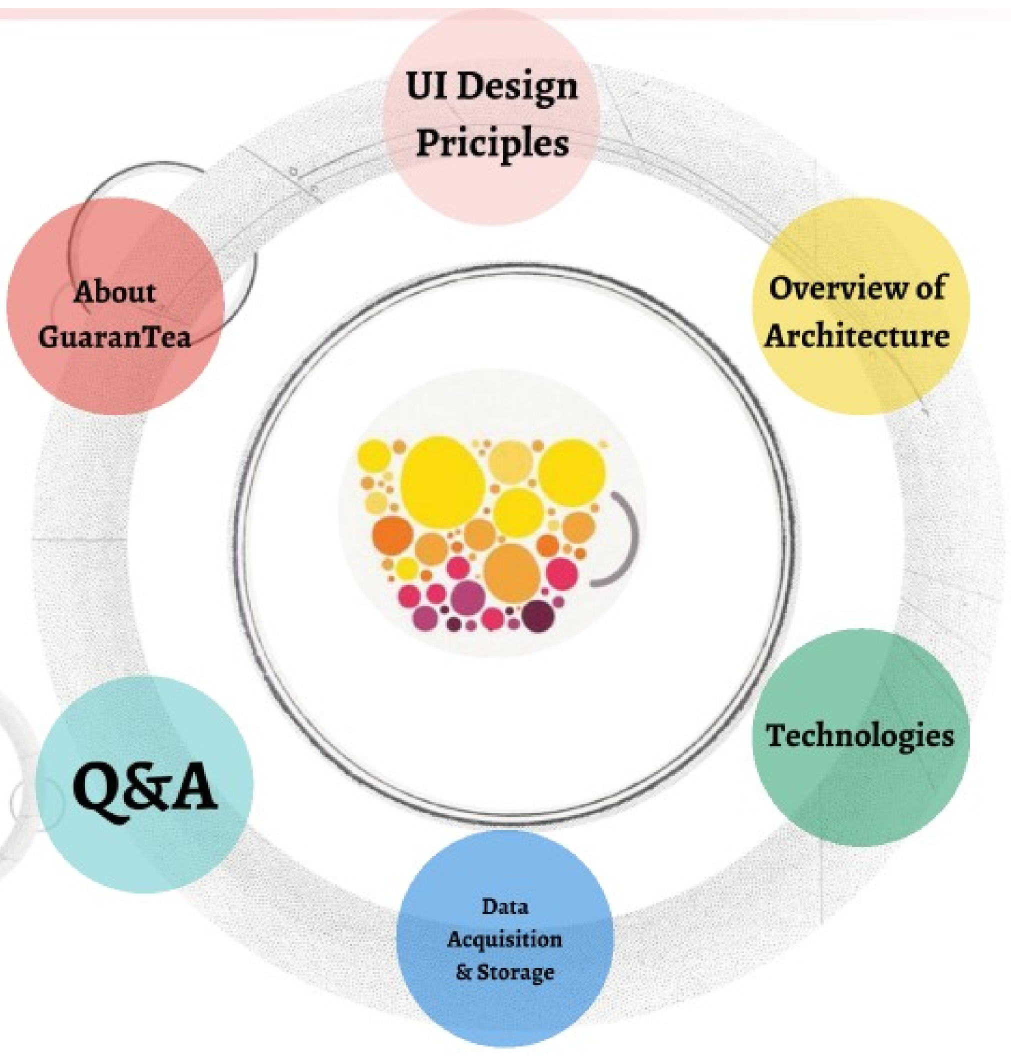
GuaranTea

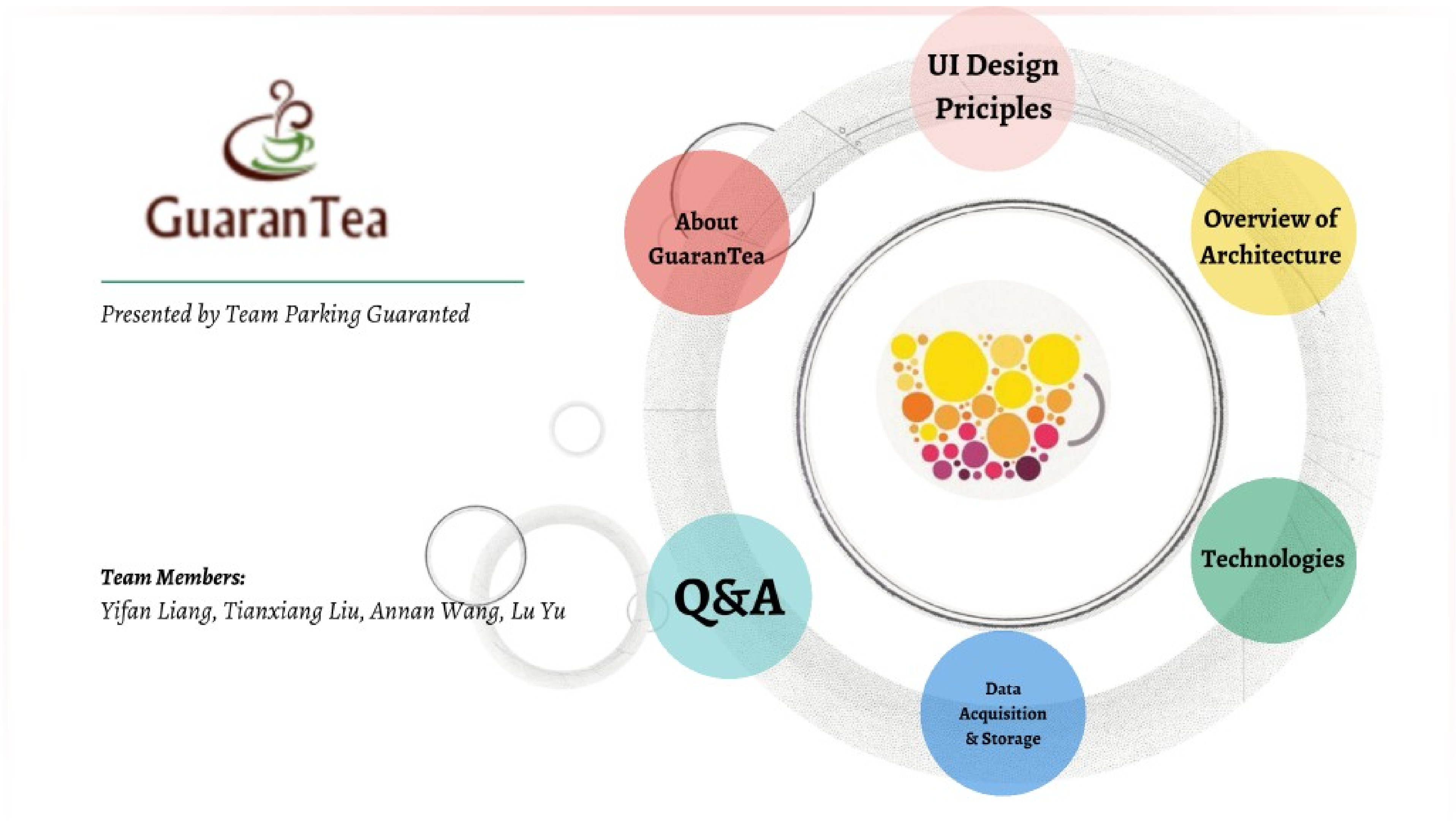
Presented by Team Parking Guaranted



Team Members:
Yifan Liang, Tianxiang Liu, Annan Wang, Lu Yu

What is GuaranTea?

An interactive online tea order website, allowing the user to order tea beverages and pick up at store.



GuaranTea

Presented by Team Parking Guaranted

Team Members:
Yifan Liang, Tianxiang Liu, Annan Wang, Lu Yu

About GuaranTea

UI Design Priciples

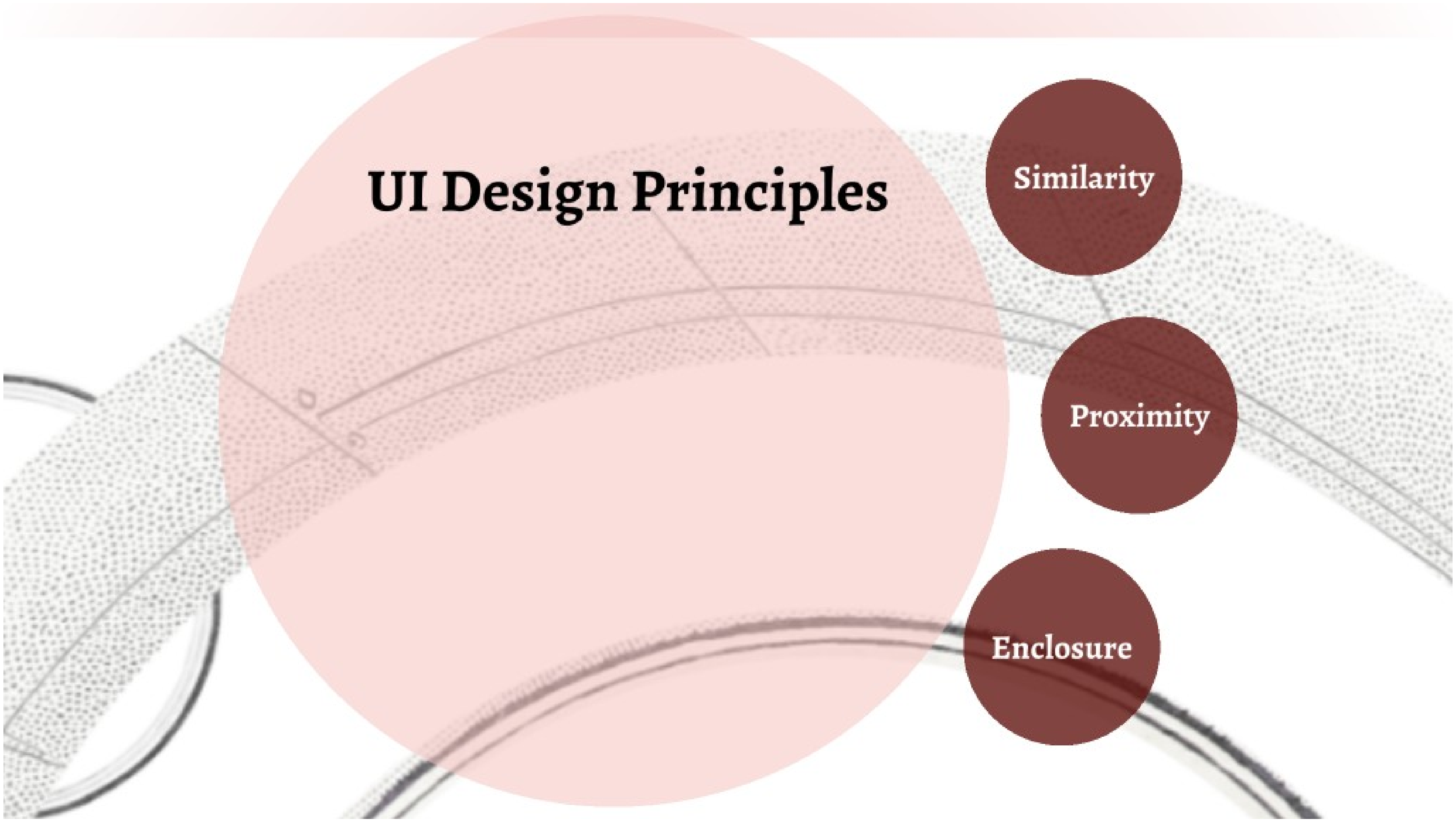
Overview of Architecture

Technologies

Data Acquisition & Storage

Q&A

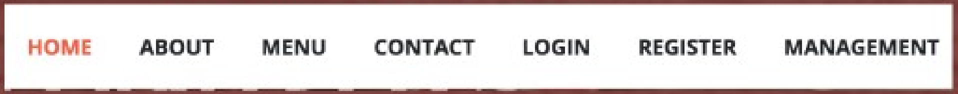
UI Design Principles



- Similarity
- Proximity
- Enclosure

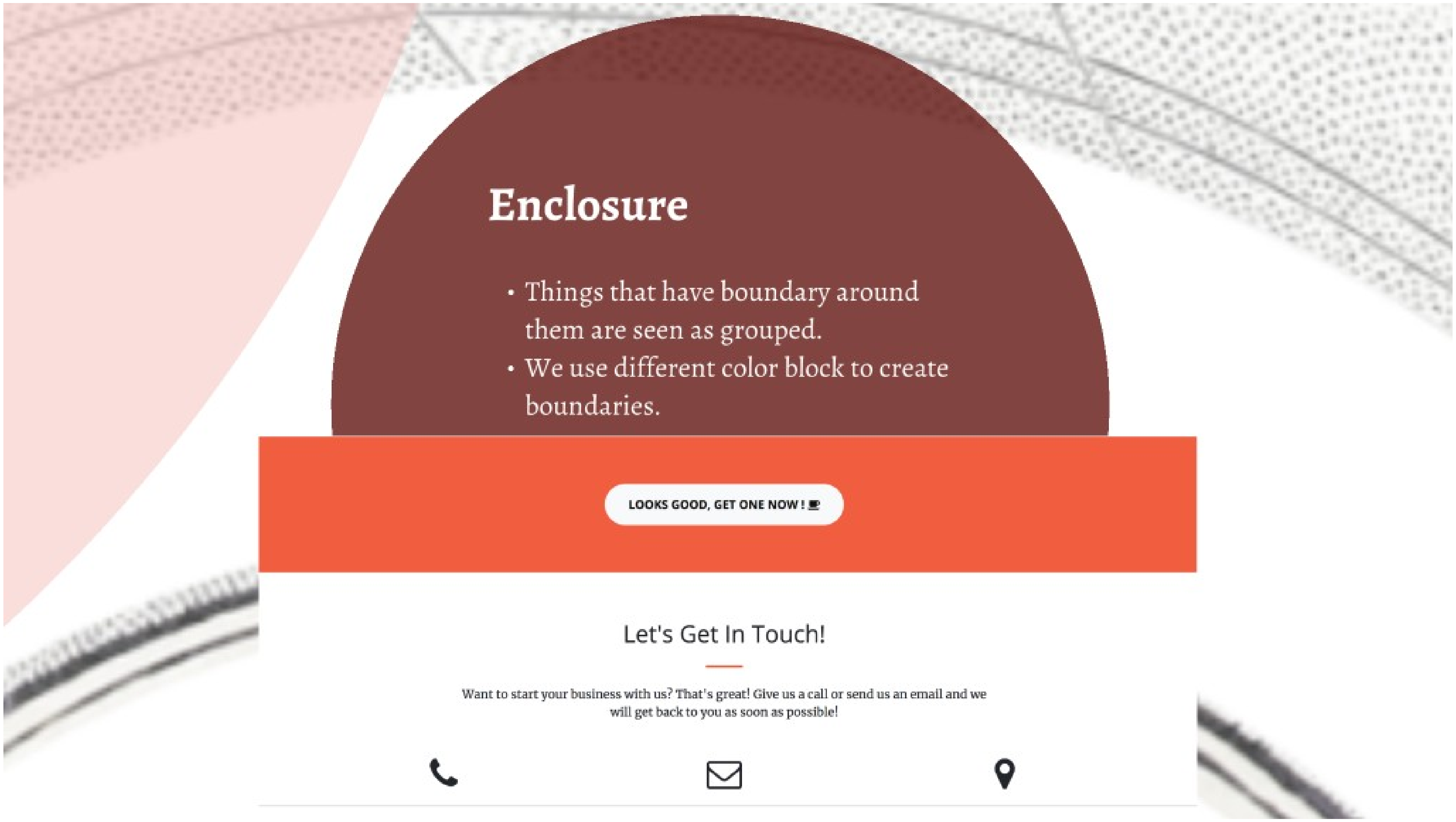
Similarity

- Buttons are similar style but not exact same in color.
- They are provided to be one group and share the same level of function.



Proximity

When elements are close together , they are seen as grouped

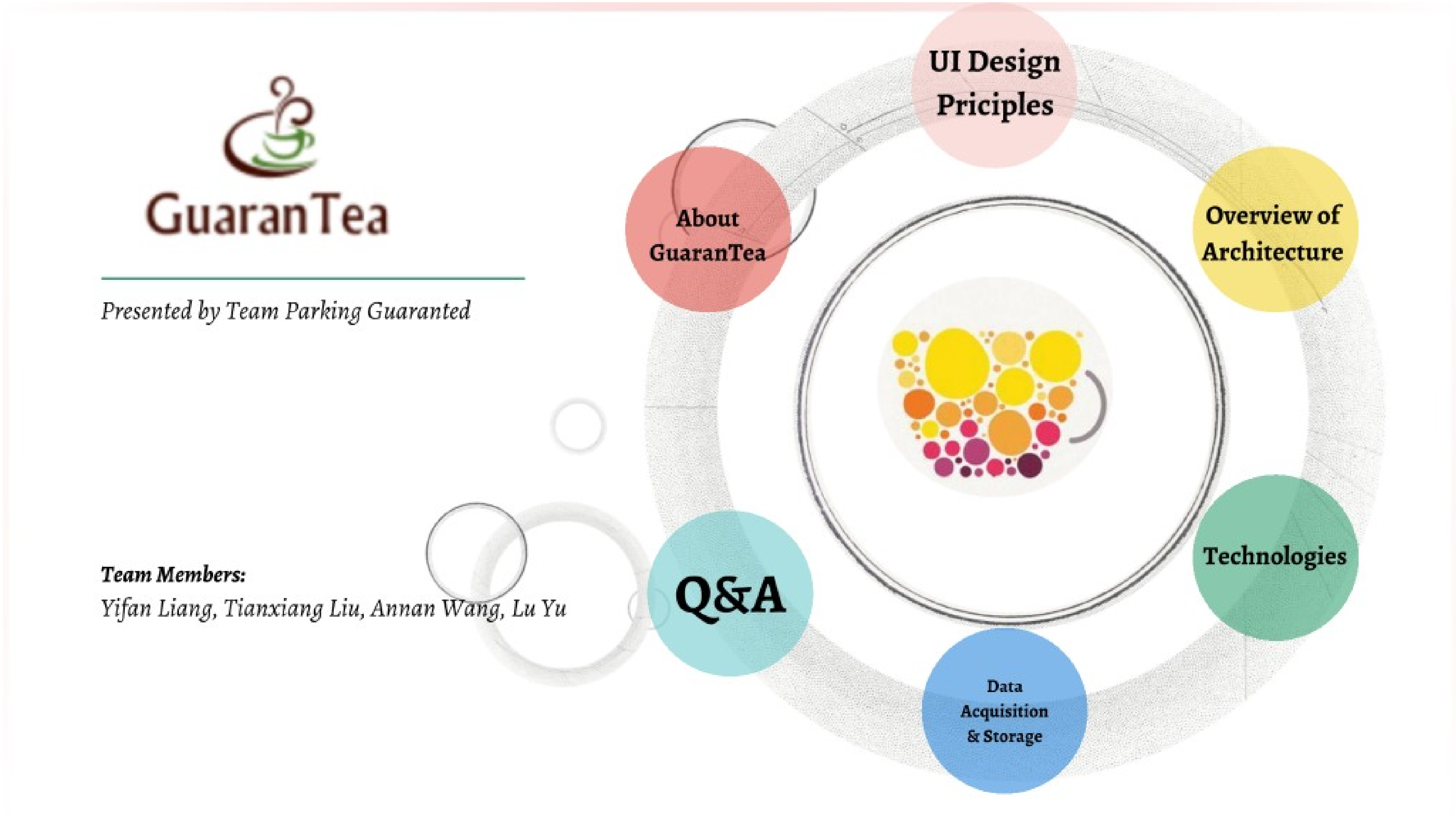


Enclosure

- Things that have boundary around them are seen as grouped.
- We use different color block to create boundaries.

Let's Get In Touch!

Want to start your business with us? That's great! Give us a call or send us an email and we will get back to you as soon as possible!



GuaranTea

Presented by Team Parking Guaranted

Team Members:
Yifan Liang, Tianxiang Liu, Annan Wang, Lu Yu

About GuaranTea

UI Design Priciples

Overview of Architecture

Technologies

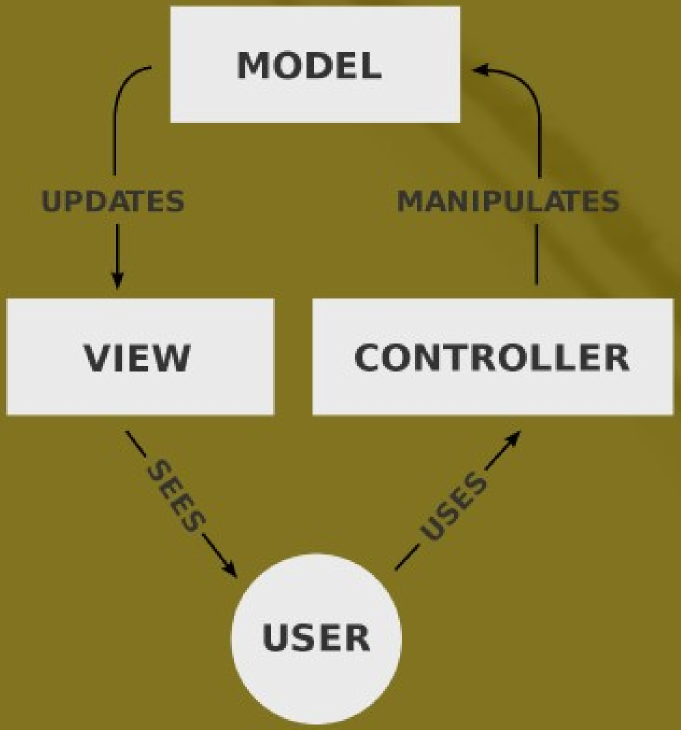
Data Acquisition & Storage

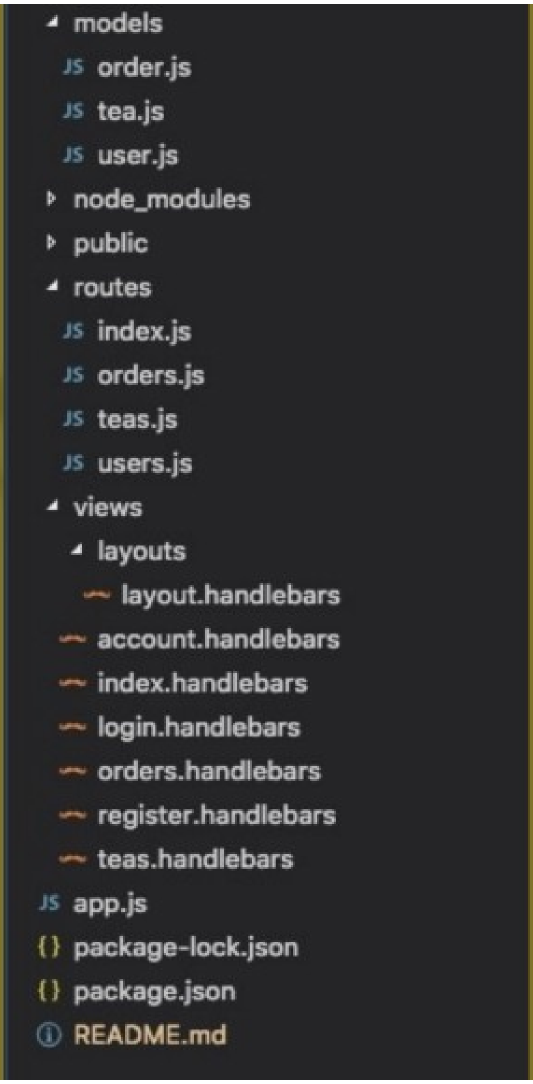
Q&A

MVC

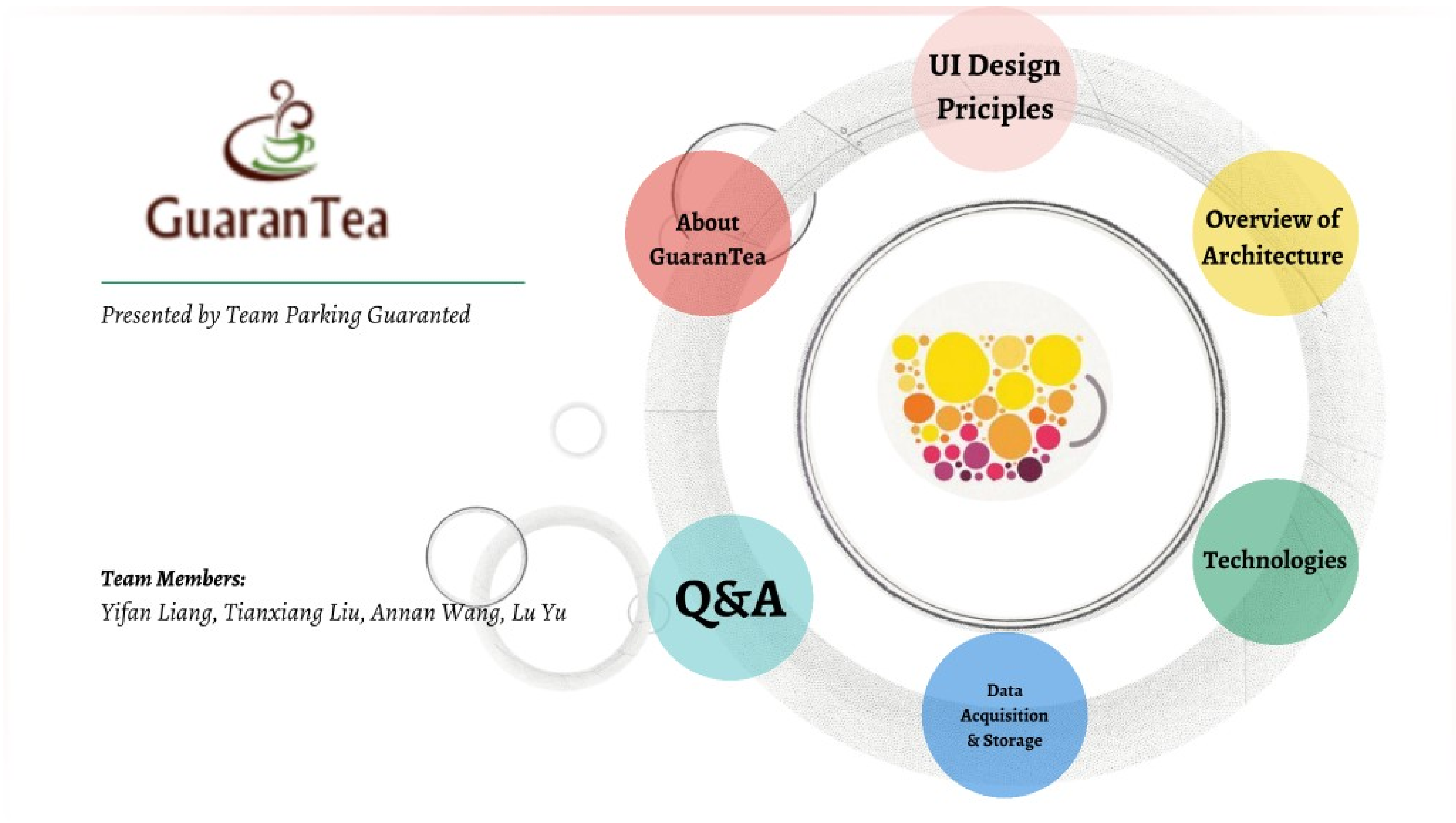
MVC Flowchart

MVC Flowchart





```
models
  order.js
  tea.js
  user.js
node_modules
public
routes
  index.js
  orders.js
  teas.js
  users.js
views
  layouts
    layout.handlebars
  account.handlebars
  index.handlebars
  login.handlebars
  orders.handlebars
  register.handlebars
  teas.handlebars
app.js
package-lock.json
package.json
README.md
```

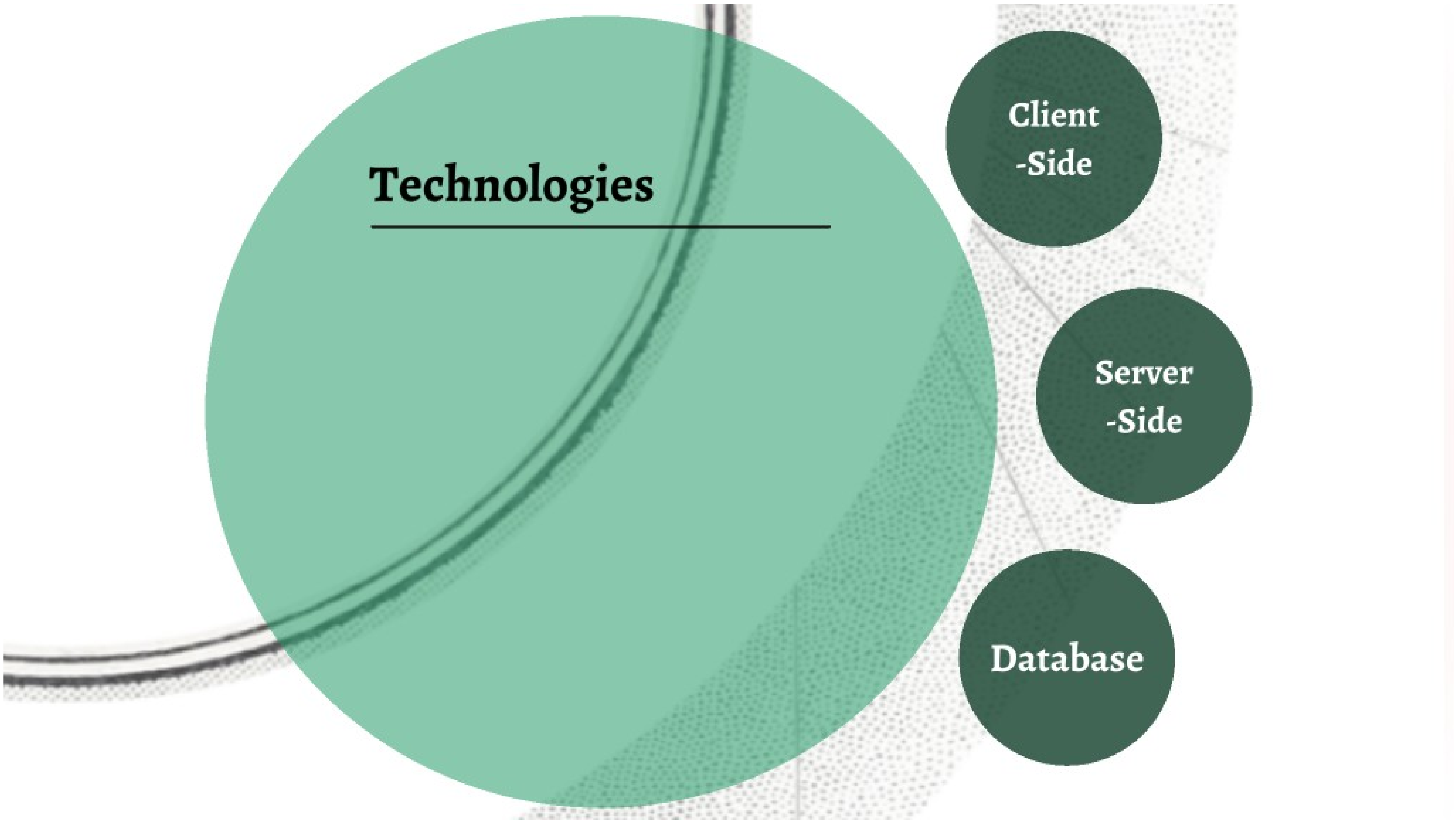


GuaranTea

Presented by Team Parking Guaranted

Team Members:
Yifan Liang, Tianxiang Liu, Annan Wang, Lu Yu

UI Design Priciples
About GuaranTea
Overview of Architecture
Q&A
Technologies
Data Acquisition & Storage

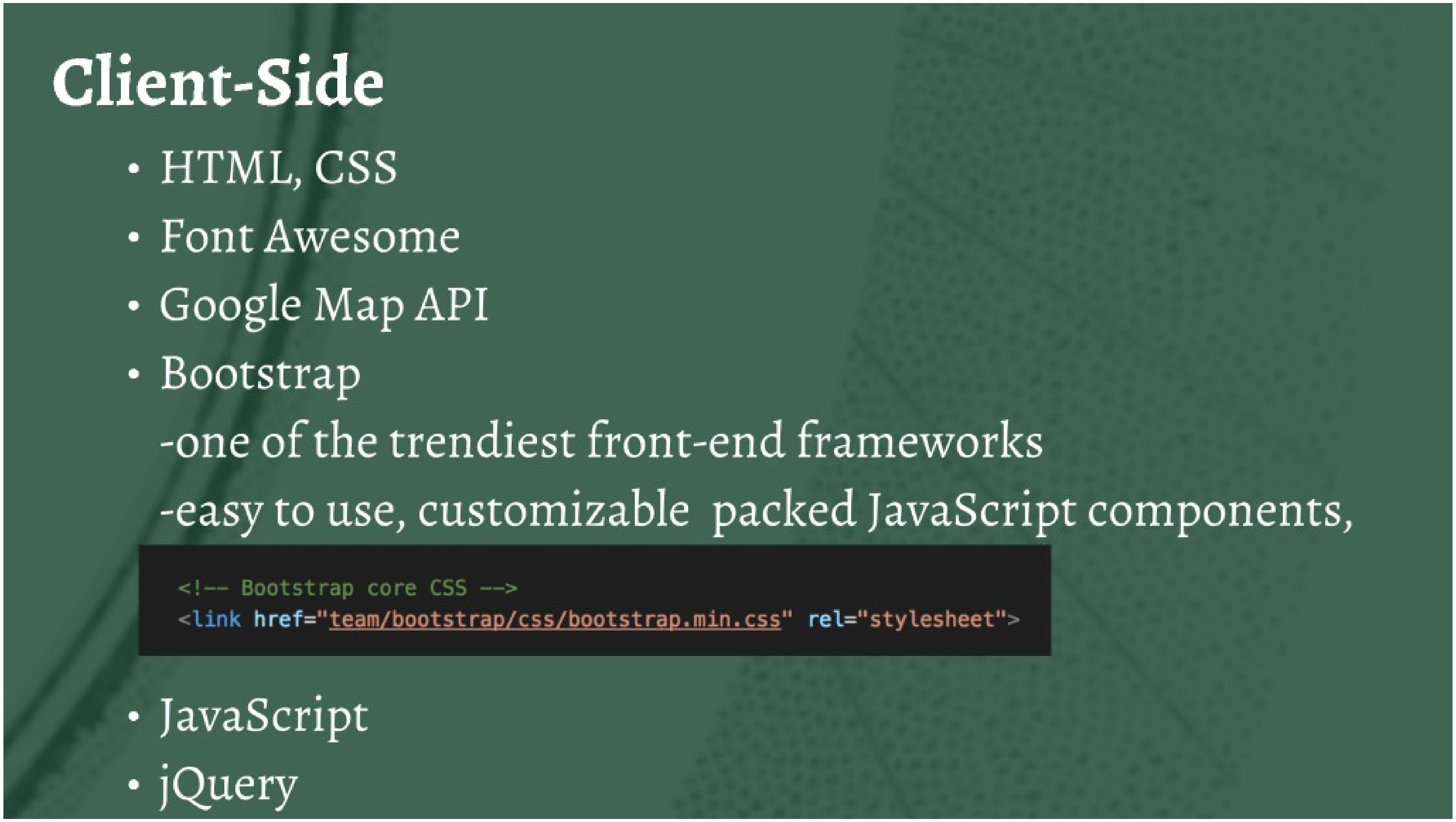


Technologies

Client -Side

Server -Side

Database



Client-Side

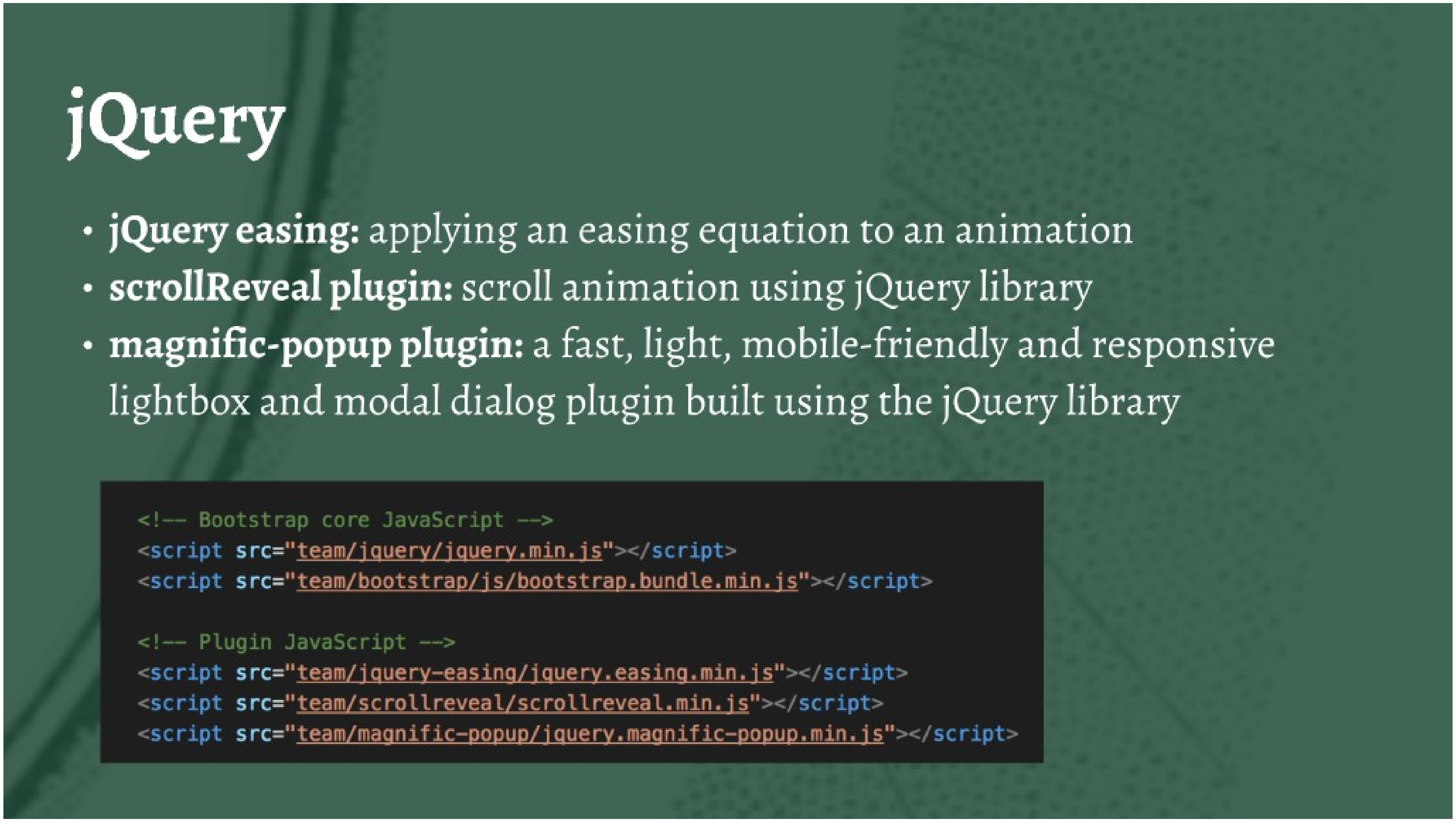
- HTML, CSS
- Font Awesome
- Google Map API
- Bootstrap

  -one of the trendiest front-end frameworks

  -easy to use, customizable packed JavaScript components,

```
<!-- Bootstrap core CSS -->
<link href="team/bootstrap/css/bootstrap.min.css" rel="stylesheet">
```

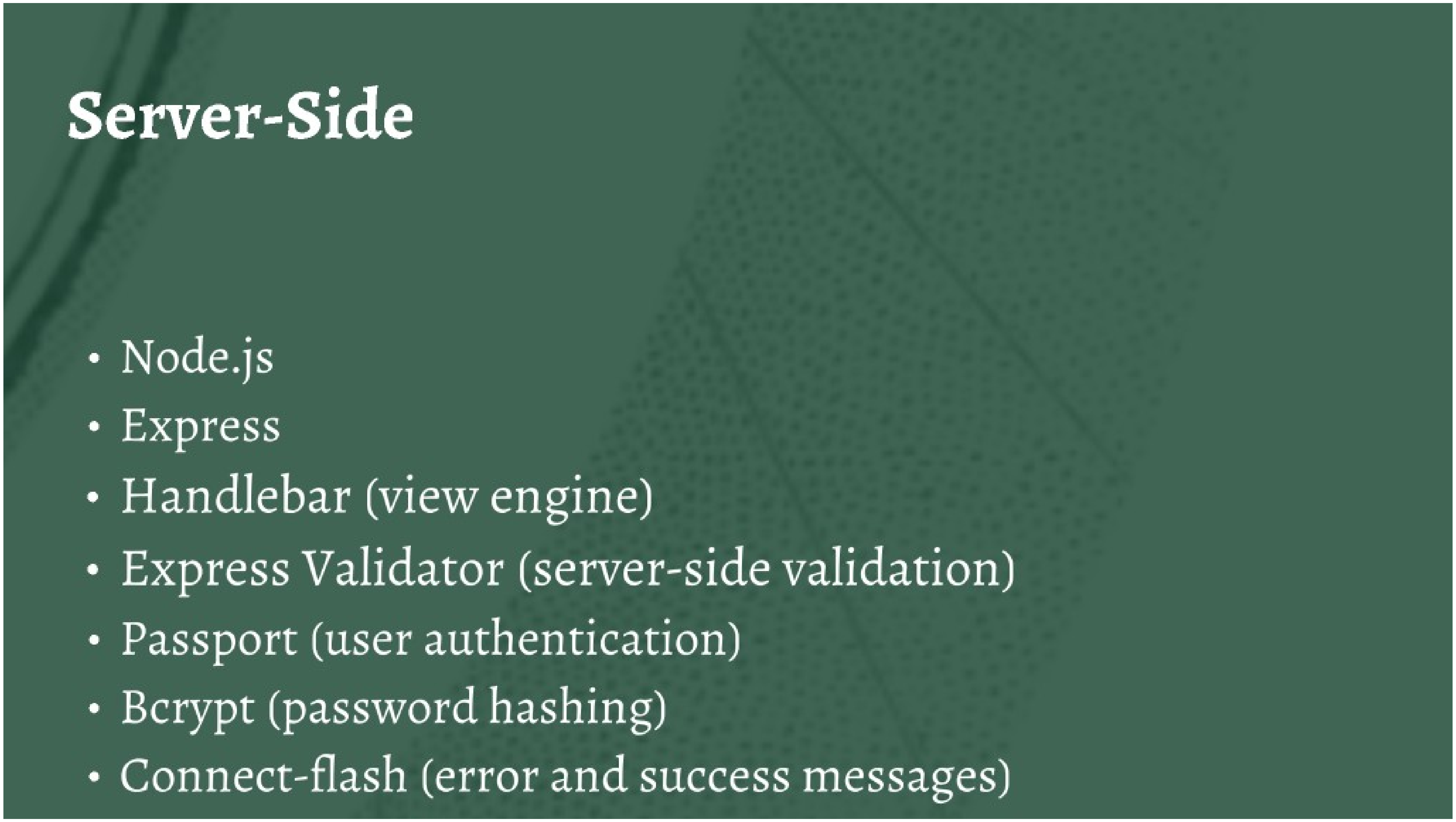
- JavaScript
- jQuery



jQuery

- **jQuery easing:** applying an easing equation to an animation
- **scrollReveal plugin:** scroll animation using jQuery library
- **magnific-popup plugin:** a fast, light, mobile-friendly and responsive lightbox and modal dialog plugin built using the jQuery library

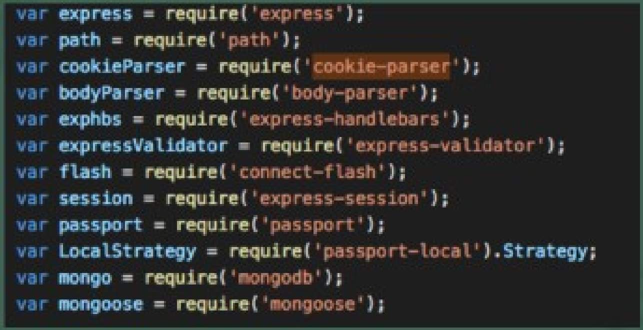
```
<!-- Bootstrap core JavaScript -->
<script src="team/jquery/jquery.min.js"></script>
<script src="team/bootstrap/js/bootstrap.bundle.min.js"></script>

<!-- Plugin JavaScript -->
<script src="team/jquery-easing/jquery.easing.min.js"></script>
<script src="team/scrollreveal/scrollreveal.min.js"></script>
<script src="team/magnific-popup/jquery.magnific-popup.min.js"></script>
```



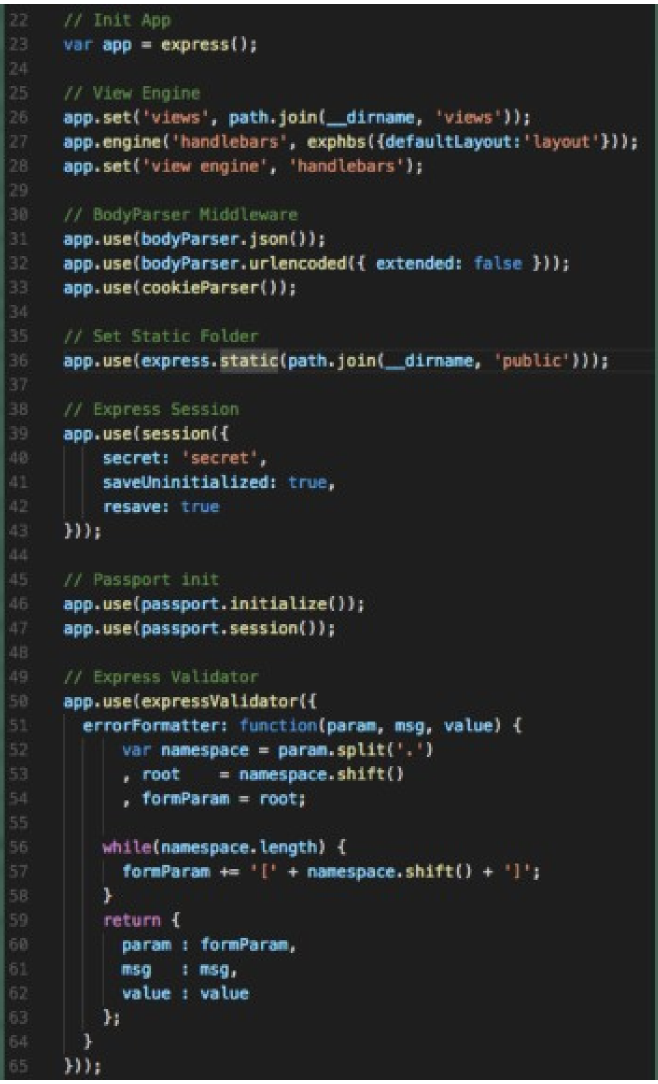
Server-Side

- Node.js
- Express
- Handlebar (view engine)
- Express Validator (server-side validation)
- Passport (user authentication)
- Bcrypt (password hashing)
- Connect-flash (error and success messages)

Import and Use Middleware

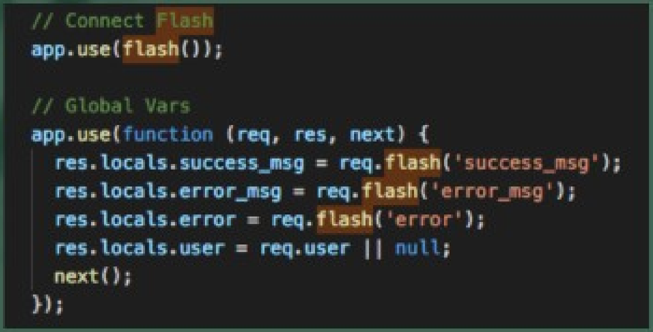


```
var express = require('express');
var path = require('path');
var cookieParser = require('cookie-parser');
var bodyParser = require('body-parser');
var exphbs = require('express-handlebars');
var expressValidator = require('express-validator');
var flash = require('connect-flash');
var session = require('express-session');
var passport = require('passport');
var LocalStrategy = require('passport-local').Strategy;
var mongo = require('mongodb');
var mongoose = require('mongoose');
```



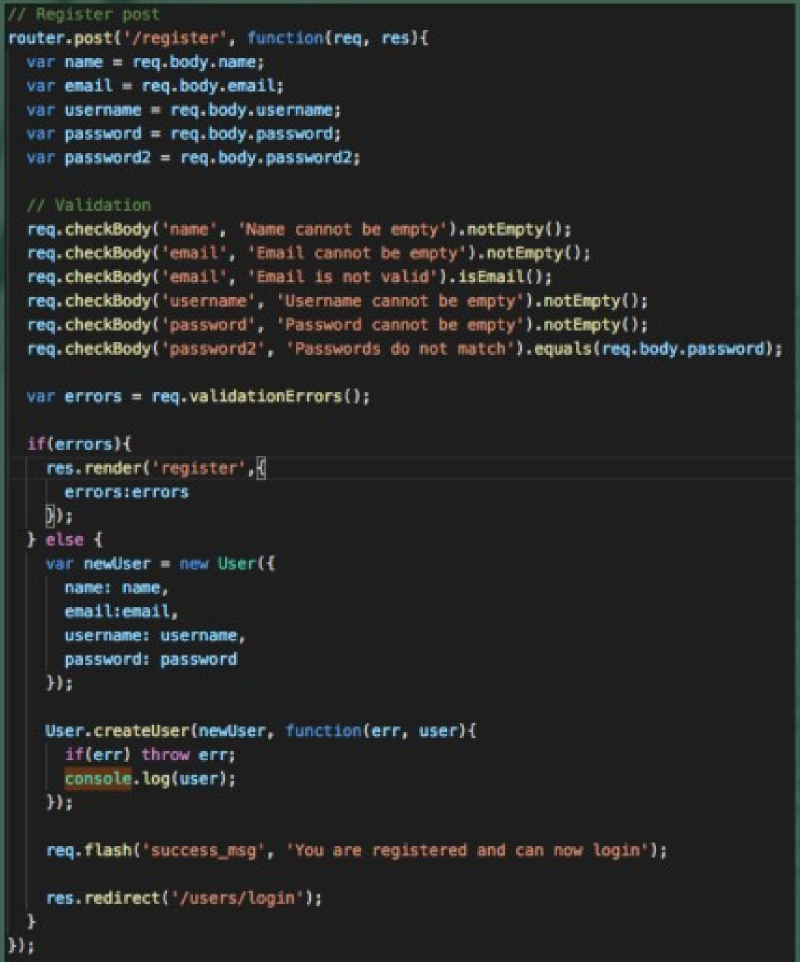
```
// Init App
var app = express();

// View Engine
app.set('views', path.join(__dirname, 'views'));
app.engine('handlebars', exphbs({defaultLayout:'layout'}));
app.set('view engine', 'handlebars');

// BodyParser Middleware
app.use(bodyParser.json());
app.use(bodyParser.urlencoded({ extended: false }));
app.use(cookieParser());

// Set Static Folder
app.use(express.static(path.join(__dirname, 'public')));

// Express Session
app.use(session({
    secret: 'secret',
    saveUninitialized: true,
    resave: true
}));

// Passport init
app.use(passport.initialize());
app.use(passport.session());

// Express Validator
app.use(expressValidator({
  errorFormatter: function(param, msg, value) {
      var namespace = param.split('.')
      , root    = namespace.shift()
      , formParam = root;

    while(namespace.length) {
      formParam += '[' + namespace.shift() + ']';
    }
    return {
      param : formParam,
      msg   : msg,
      value : value
    };
  }
}));
```

Connect-flash (error and seccess messages)



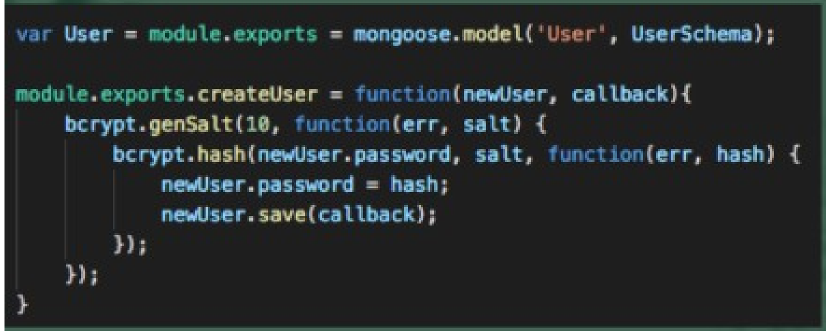
```
// Connect Flash
app.use(flash());

// Global Vars
app.use(function (req, res, next) {
  res.locals.success_msg = req.flash('success_msg');
  res.locals.error_msg = req.flash('error_msg');
  res.locals.error = req.flash('error');
  res.locals.user = req.user || null;
  next();
});
```

Express Validator (server-side validation)



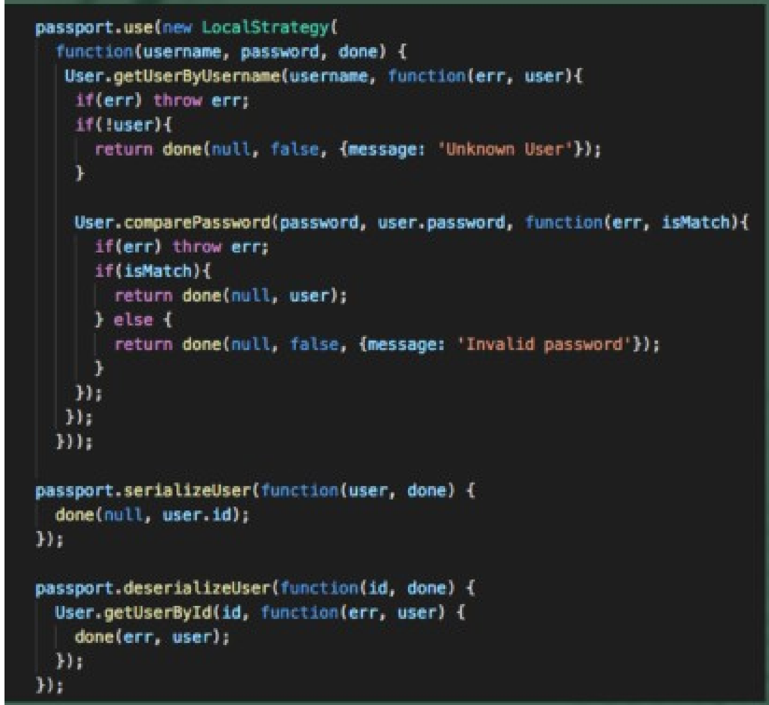
```
// Register post
router.post('/register', function(req, res){
  var name = req.body.name;
  var email = req.body.email;
  var username = req.body.username;
  var password = req.body.password;
  var password2 = req.body.password2;

  // Validation
  req.checkBody('name', 'Name cannot be empty').notEmpty();
  req.checkBody('email', 'Email cannot be empty').notEmpty();
  req.checkBody('email', 'Email is not valid').isEmail();
  req.checkBody('username', 'Username cannot be empty').notEmpty();
  req.checkBody('password', 'Password cannot be empty').notEmpty();
  req.checkBody('password2', 'Passwords do not match').equals(req.body.password);

  var errors = req.validationErrors();

  if(errors){
    res.render('register',{
      errors:errors
    });
  } else {
    var newUser = new User({
      name: name,
      email:email,
      username: username,
      password: password
    });

    User.createUser(newUser, function(err, user){
      if(err) throw err;
      console.log(user);
    });

    req.flash('success_msg', 'You are registered and can now login');

    res.redirect('/users/login');
  }
});
```

Bcrypt (password hashing)



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

module.exports.createUser = function(newUser, callback){
    bcrypt.genSalt(10, function(err, salt) {
        bcrypt.hash(newUser.password, salt, function(err, hash) {
            newUser.password = hash;
            newUser.save(callback);
        });
    });
}
```

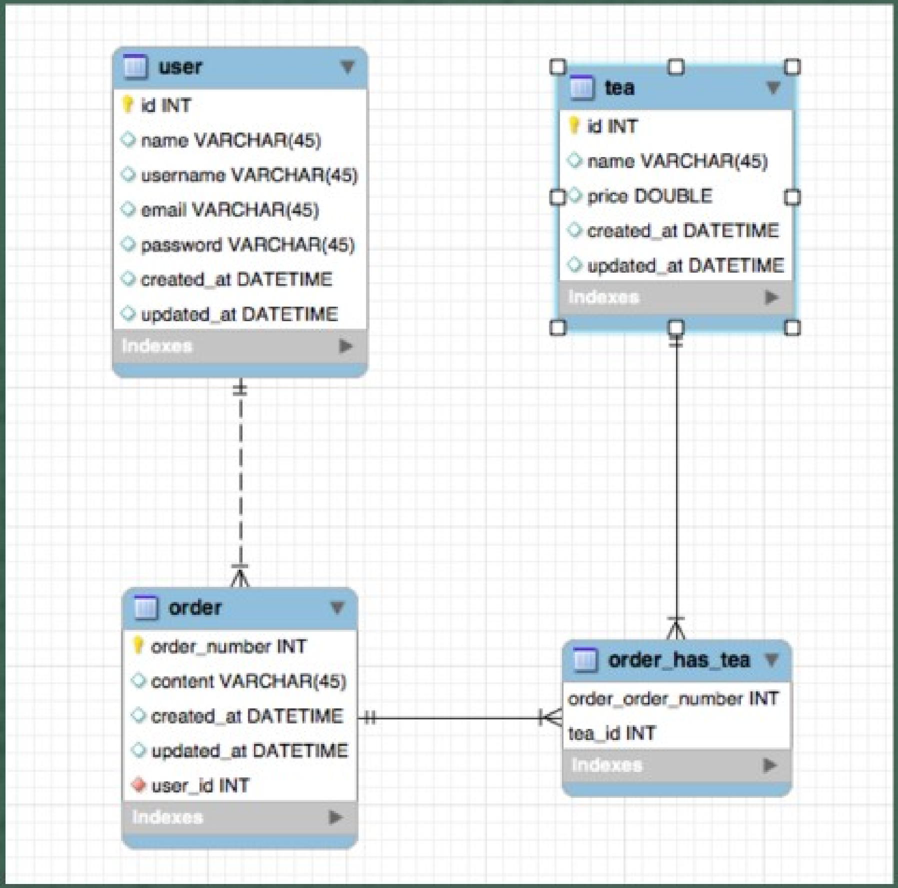
Passport (user authentication)

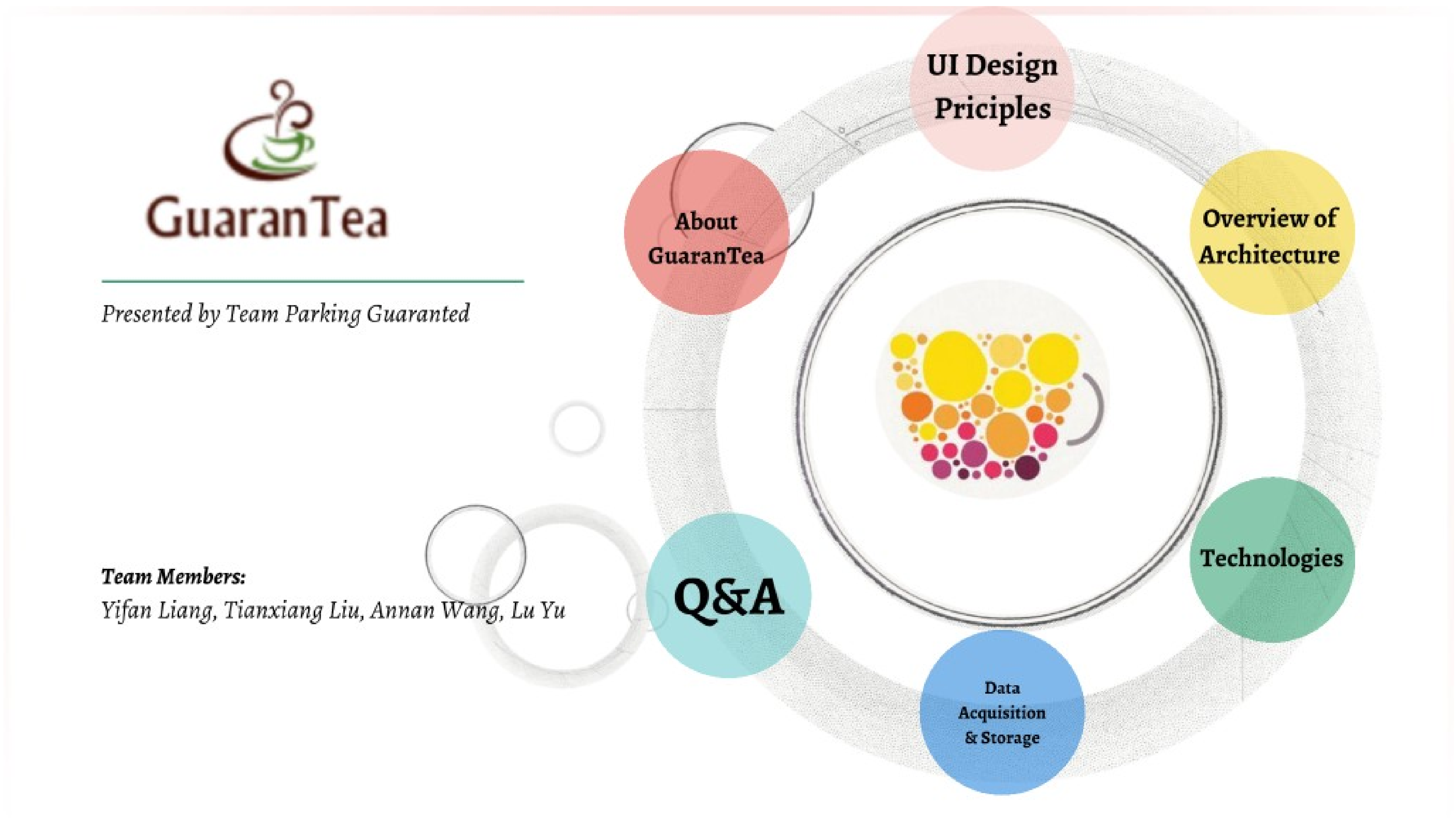


```
passport.use(new LocalStrategy(
  function(username, password, done) {
   User.getUserByUsername(username, function(err, user){
    if(err) throw err;
    if(!user){
      return done(null, false, {message: 'Unknown User'});
    }

    User.comparePassword(password, user.password, function(err, isMatch){
      if(err) throw err;
      if(isMatch){
        return done(null, user);
      } else {
        return done(null, false, {message: 'Invalid password'});
      }
    });
   });
  }));

passport.serializeUser(function(user, done) {
  done(null, user.id);
});

passport.deserializeUser(function(id, done) {
  User.getUserById(id, function(err, user) {
    done(err, user);
  });
});
```

Database

- Mongo DB
- Mongoose





GuaranTea

Presented by Team Parking Guaranted

Team Members:
Yifan Liang, Tianxiang Liu, Annan Wang, Lu Yu

UI Design Priciples
About GuaranTea
Overview of Architecture
Technologies
Data Acquisition & Storage
Q&A

Data Acquisition & Storage

- The website will get data from users and store owners.
- After that, it will store the data to mongoDB.

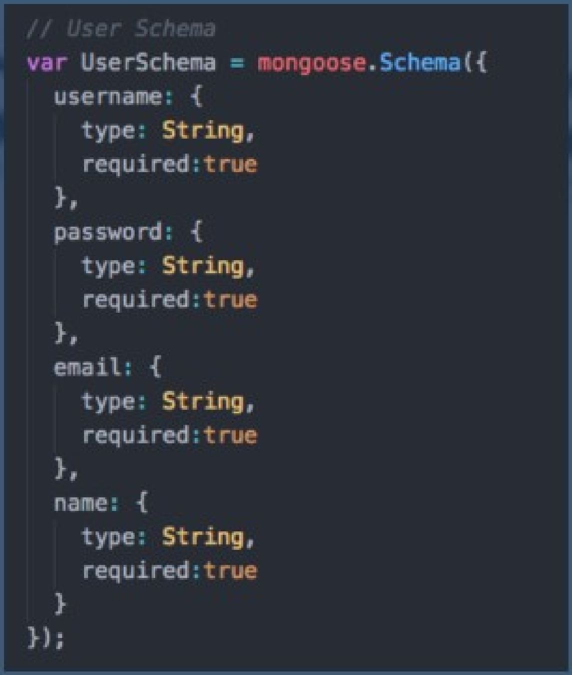
User Data

Tea Beverage Data

Order Data

User Data

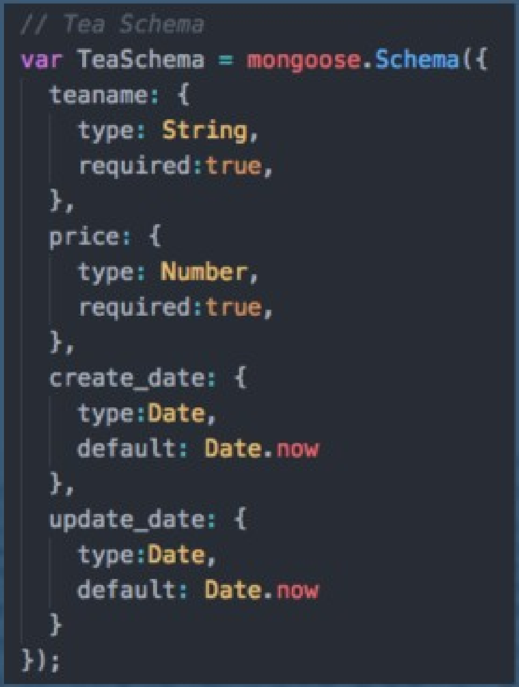
- User uploads name, username, password and email address upon registration.



```
// User Schema
var UserSchema = mongoose.Schema({
  username: {
    type: String,
    required:true
  },
  password: {
    type: String,
    required:true
  },
  email: {
    type: String,
    required:true
  },
  name: {
    type: String,
    required:true
  }
});
```

Tea Beverage Data

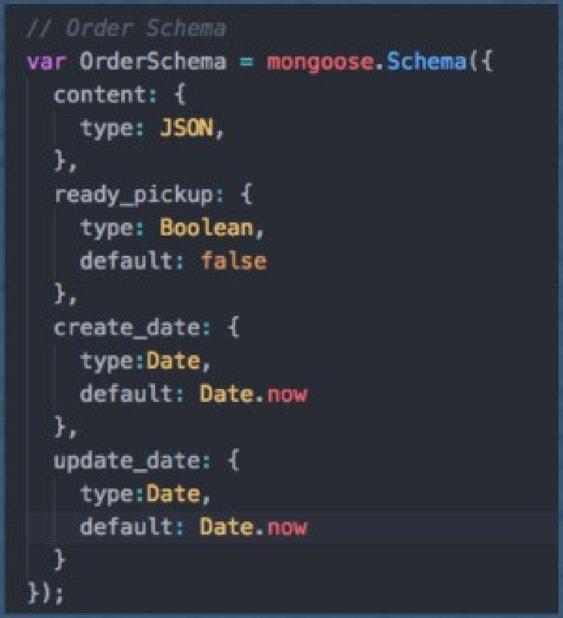
- Bussiness owner can upload beverage name and price.



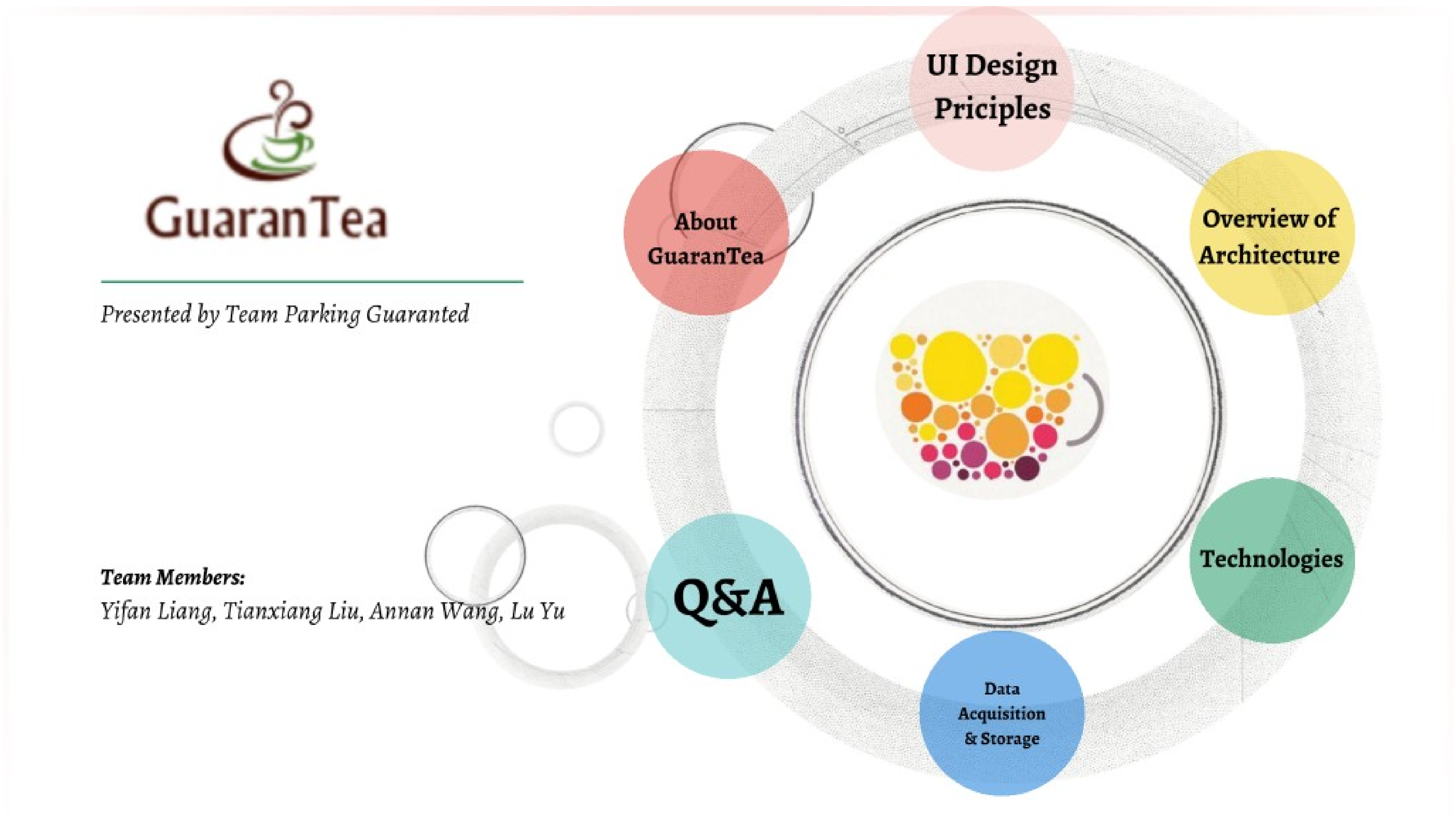
```
// Tea Schema
var TeaSchema = mongoose.Schema({
  teaname: {
    type: String,
    required:true,
  },
  price: {
    type: Number,
    required:true,
  },
  create_date: {
    type:Date,
    default: Date.now
  },
  update_date: {
    type:Date,
    default: Date.now
  }
});
```

Order Data

- The website keeps track of user's order information including tea name, price and when the order was placed.



```
// Order Schema
var OrderSchema = mongoose.Schema({
  content: {
    type: JSON,
  },
  ready_pickup: {
    type: Boolean,
    default: false
  },
  create_date: {
    type:Date,
    default: Date.now
  },
  update_date: {
    type:Date,
    default: Date.now
  }
});
```



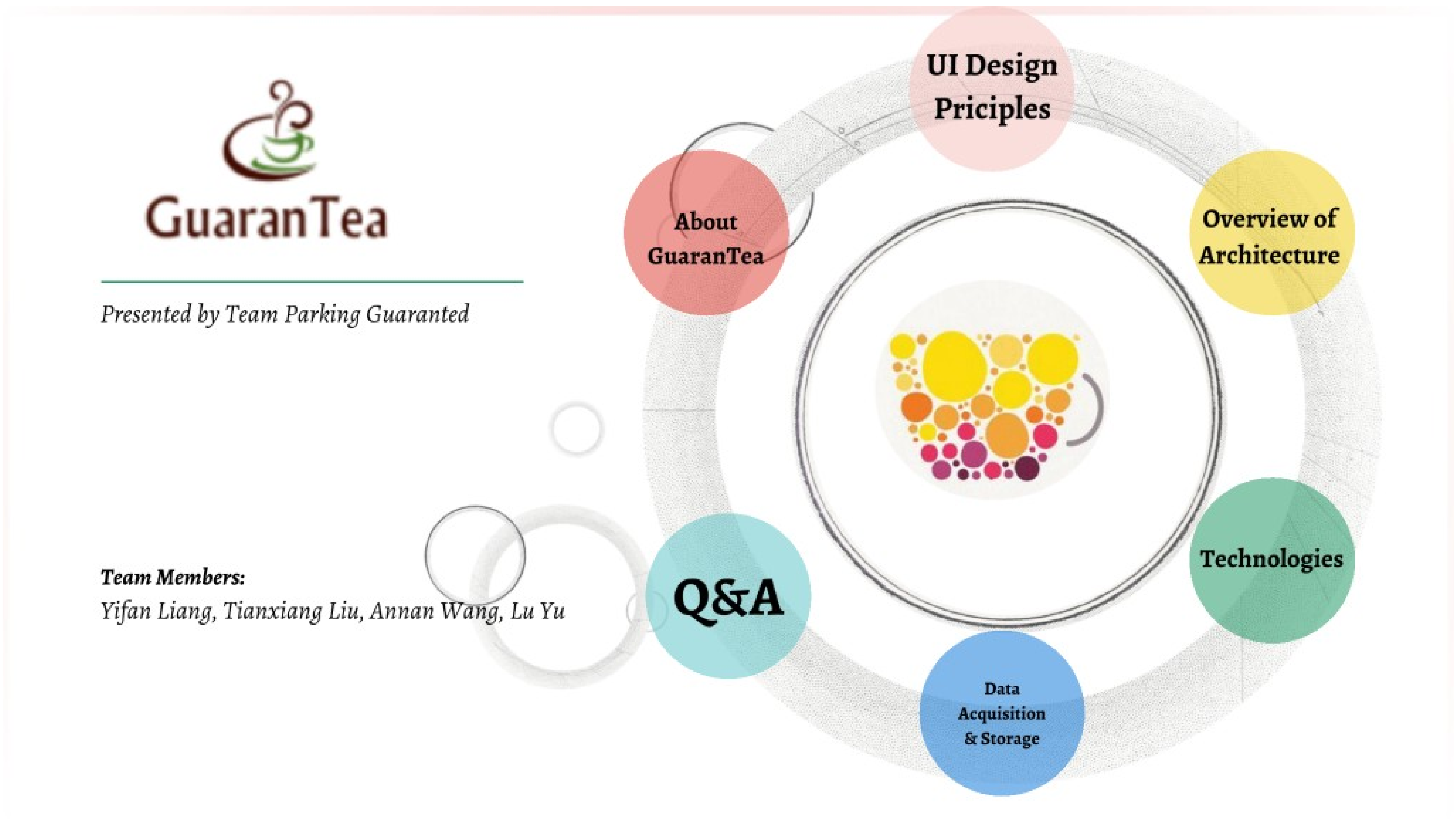
GuaranTea

Presented by Team Parking Guaranted

Team Members:
Yifan Liang, Tianxiang Liu, Annan Wang, Lu Yu

Q&A

Any questions?



GuaranTea

Presented by Team Parking Guaranted

Team Members:
Yifan Liang, Tianxiang Liu, Annan Wang, Lu Yu

UI Design Priciples
About GuaranTea
Overview of Architecture
Technologies
Data Acquisition & Storage
Q&A