



CSE370 : Database Systems

Project Report

Project Title : Marriage Registration Portal

Group No : 11, CSE370 Lab Section : 04, Summer 2023

ID	Name	Contribution
21301284	Khawaja Ehsun Ul Hawak	Everything

Table of Contents

Section No	Content	Page No
1	Introduction	3
2	Project Features	3
3	ER/EER Diagram	4
4	Schema Diagram	5
5	Frontend Development	6
6	Backend Development	11
7	Conclusion	16
8	References	16

Introduction

Introducing our advanced Marriage Registration Portal, designed to streamline the marriage process. In today's fast-paced world, couples deserve an efficient and easy way to make their unions official. This report delves into the reasons behind creating the portal, highlighting its crucial role in updating essential legal documentation. It finds a balance between tradition and innovation, providing a user-friendly digital platform for registering marriages. Beyond convenience, accurate registration has significant legal and societal implications, safeguarding spousal rights and ensuring the authenticity of family ties. The report underscores the portal's ability to lessen administrative burdens, minimize errors, and improve transparency—a step towards effective governance and the protection of individual rights.

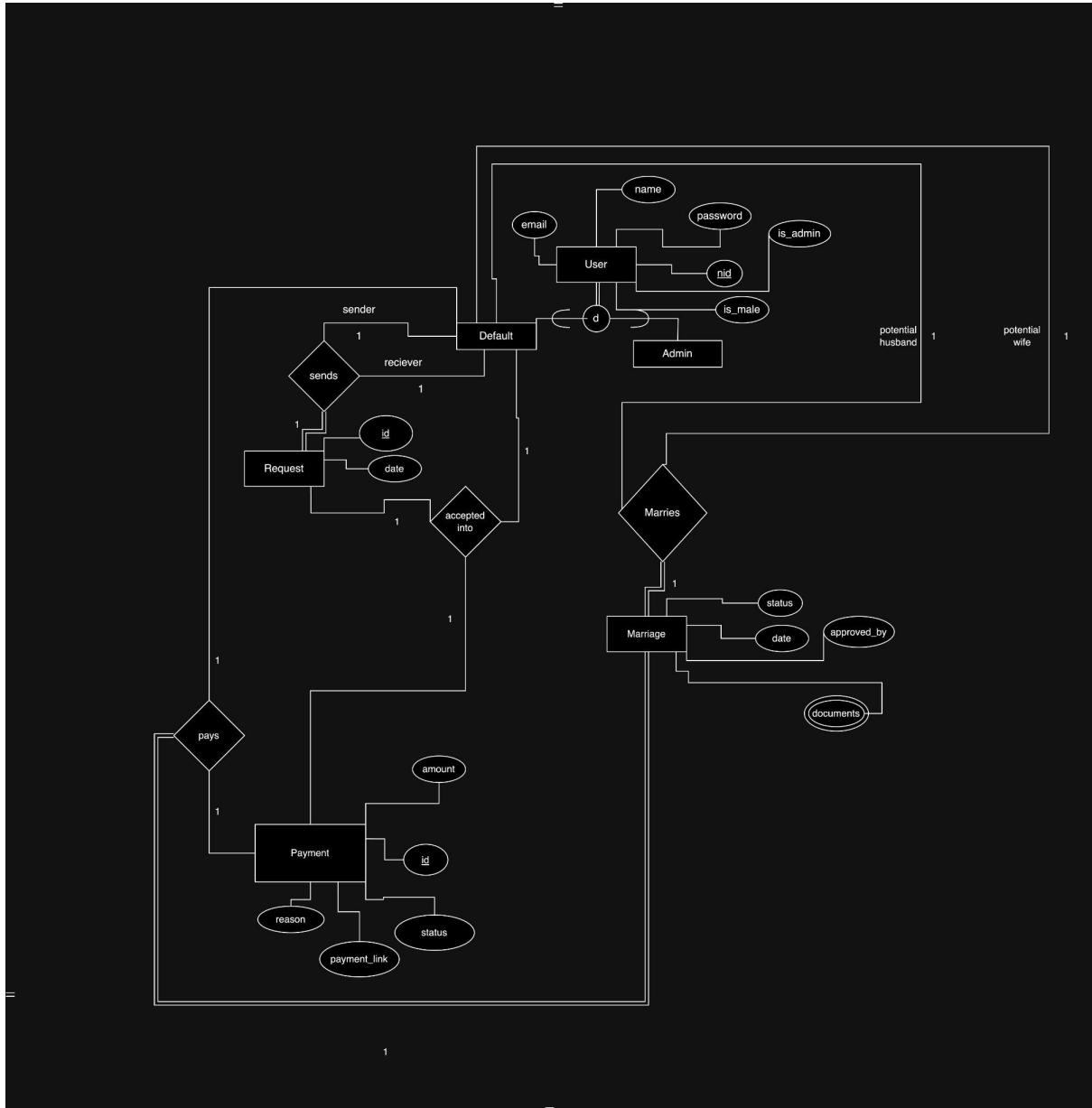
Project Features

The web application contains features necessary to initiate and divorce marriages and everything in between.

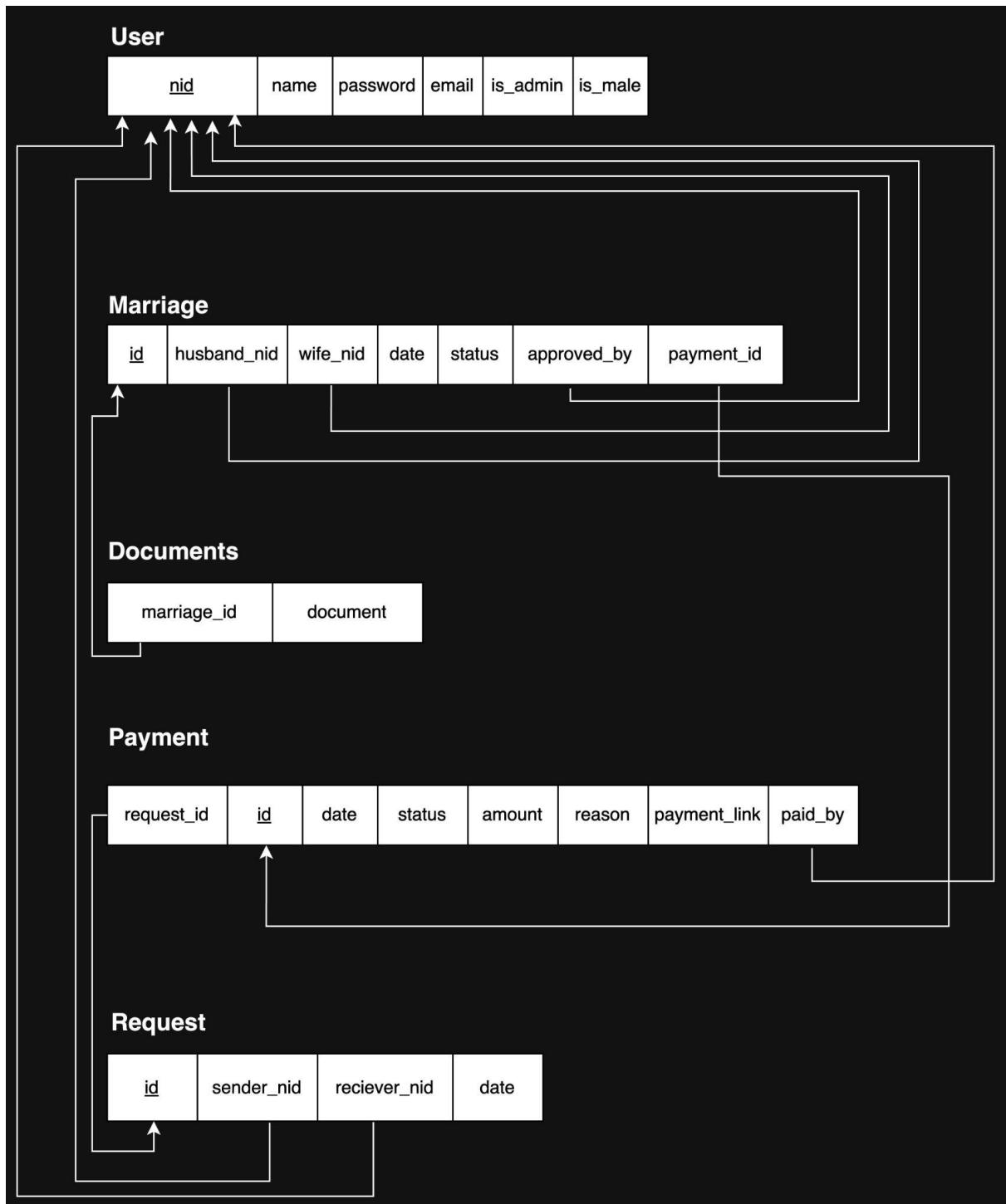
The features are listed below:

1. **Authentication:** Each citizen of the country can register their account on the portal using their nid number to be available for marriage requests. They can later login with the password they set in the registration.
2. **Marriage Requests:** Marriages requests are like friend requests that can be send by both the male and female citizens to each other(if both of them do not have a partner). Once a request is made, the other party will have to accept the request and proceed with the necessary payment to advance the request into an incomplete marriage.
3. **Marriages:** A marriage starts as an incomplete marriage. ‘Incomplete’ implies that the marriage is still not complete and has not been submitted to the admins for approval. To complete it, one has to upload the necessary documents from the portal. After the documents have been submitted, the marriage status changes into unapproved.
4. **Admin:** Admins have two tasks, one is to approve or reject the marriages that are awaiting approval, other is to initiate the divorce when ordered from court. Users with admin access can easily see the documents and approve/reject the marriage. They can also search for marriages using the nid of husband and wife and change their marriages status to divorce.
5. **Payments:** As discussed in other features, payments are a crucial part and feature of the application.

ER/EER Diagram



Schema Diagram

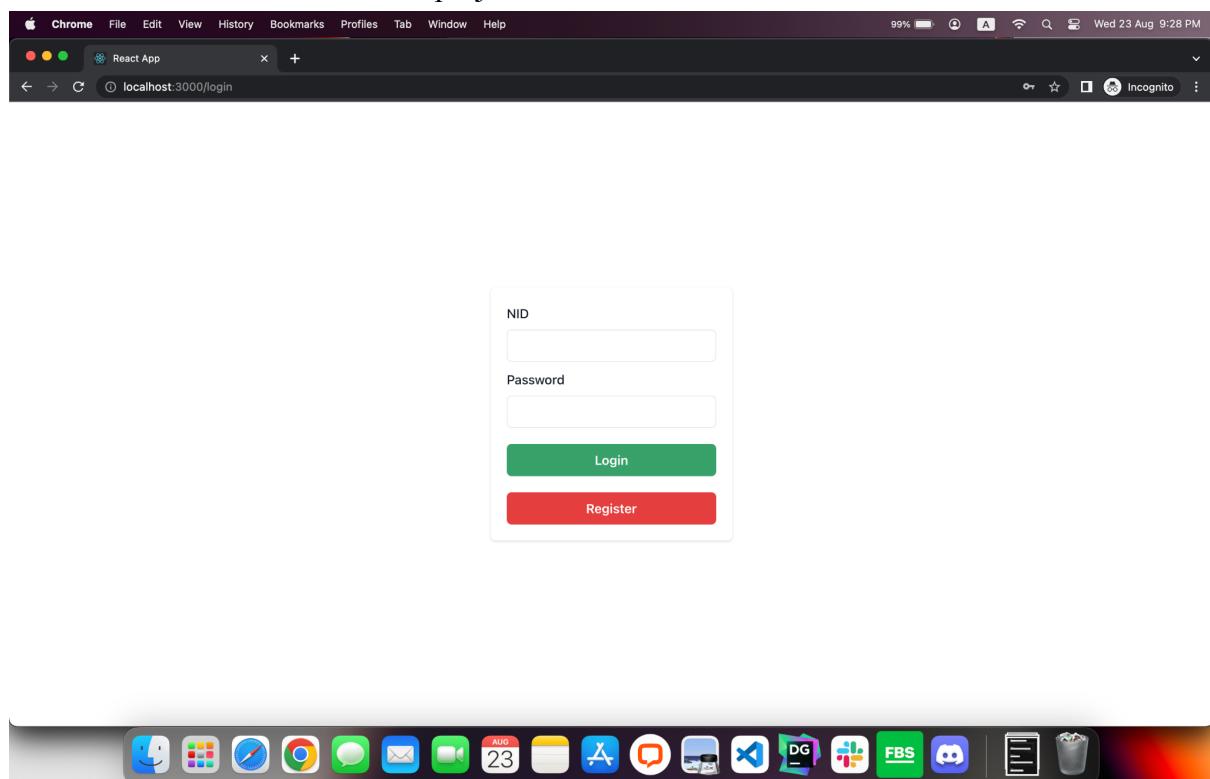


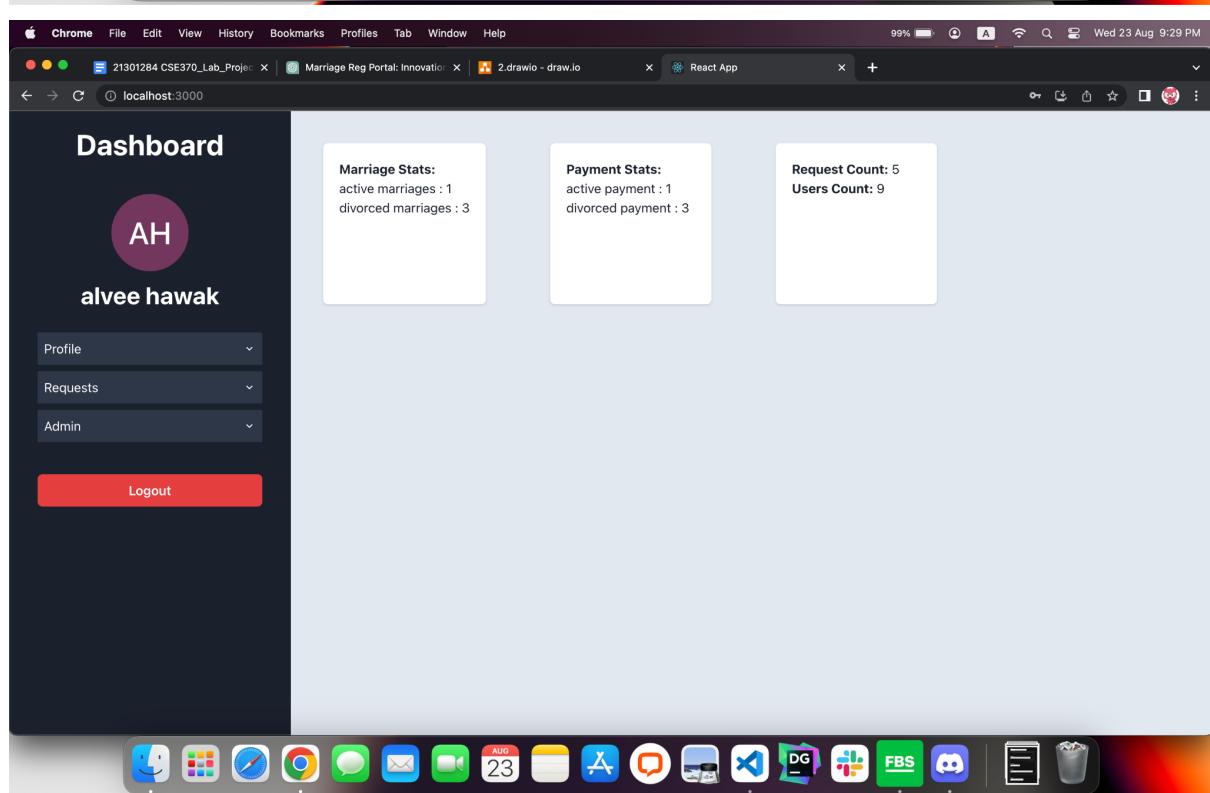
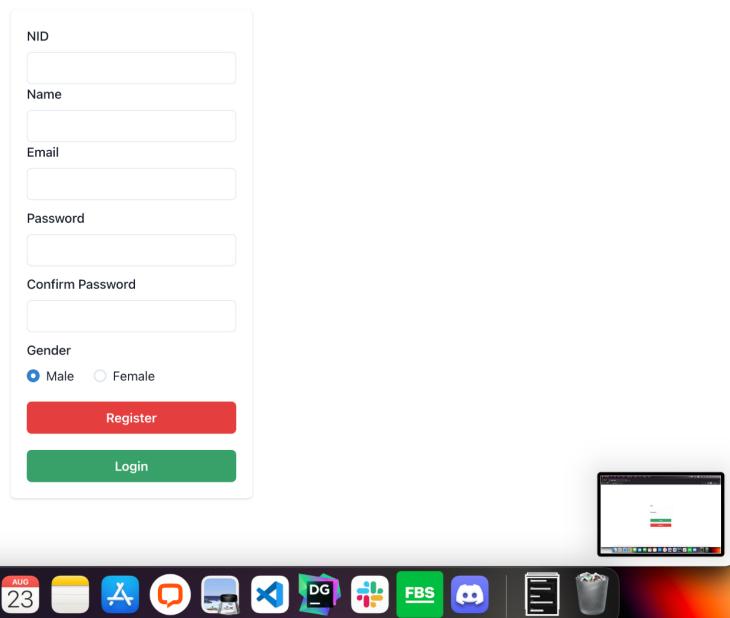
Frontend Development

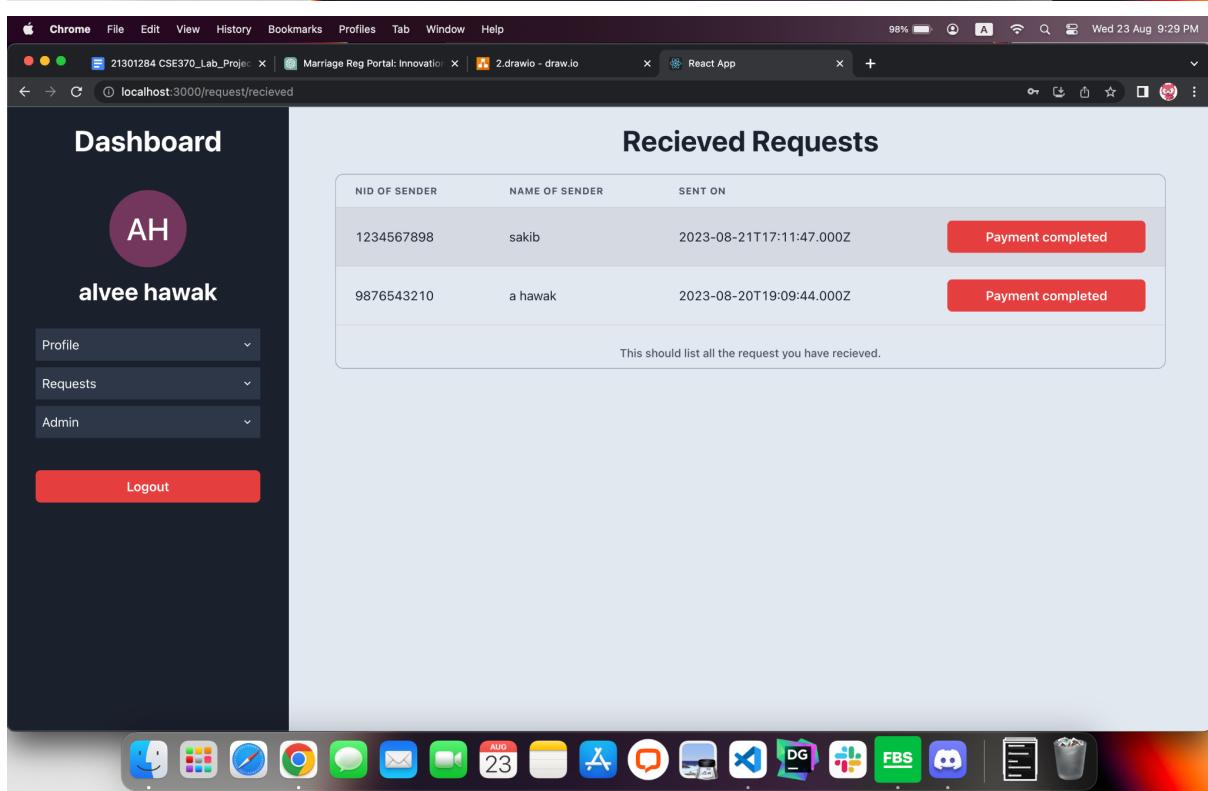
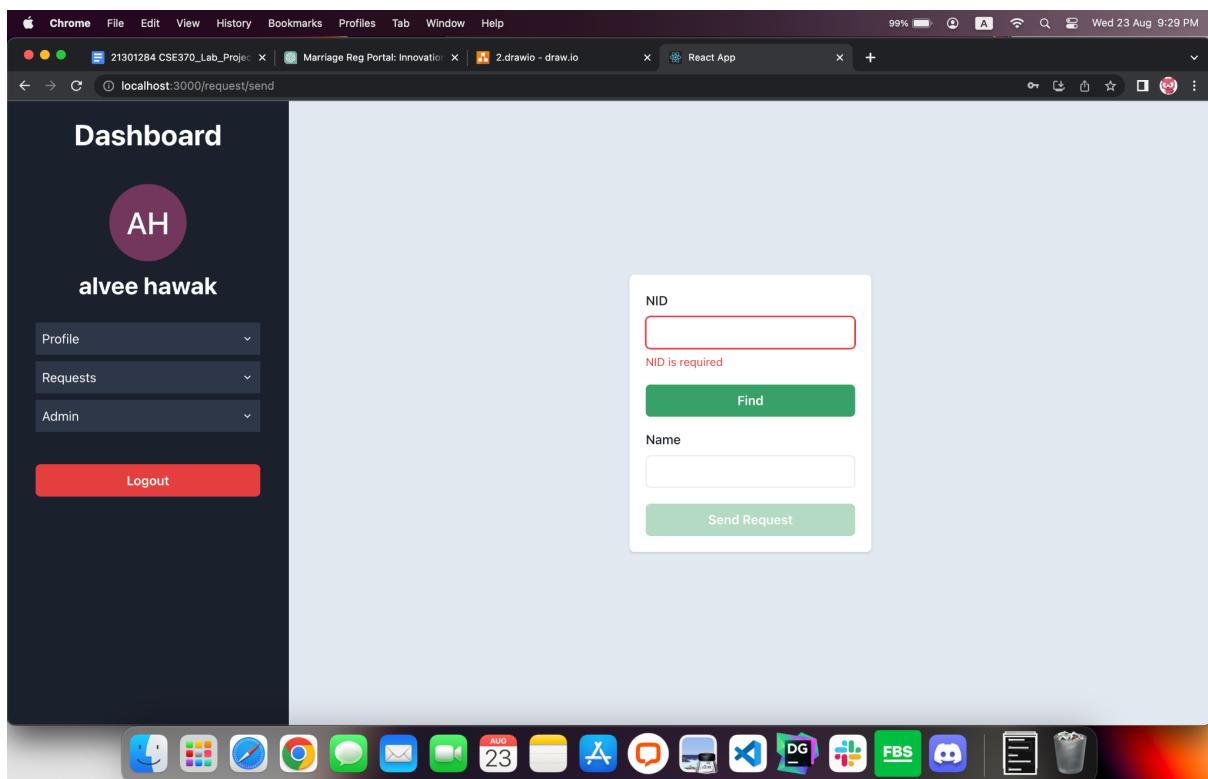
Frontend has been coded in javascript using react.js, chakra ui and other nodejs libraries.

Separate pages has been coded for login, registration, sending requests, seeing all the received requests(payments are paid in this screen aswell), checking all the marriages of the users(uploading documents are done here aswell), 2 admin pages to do divorce and approving marriages.

Here are some screenshots of the project's frontend:







Chrome File Edit View History Bookmarks Profiles Tab Window Help

21301284 CSE370_Lab_Project | Marriage Reg Portal: Innovation | 2.drawio - draw.io | React App

localhost:3000/marriages

Wednesday, Aug 23, 9:29 PM

Dashboard

AH
alvee hawak

Profile Requests Admin

Logout

Marriages

NAME OF HUSBAND	NID OF HUSBAND	NAME OF WIFE	NID OF WIFE	STATUS
alvee hawak	1234567890	a hawak	9876543210	active
alvee hawak	1234567890	sakib	1234567898	divorced

This should list all of your marriages.



Dashboard

AH

alvee hawak

Profile Requests Admin

Logout

Awaiting Approval

NID OF HUSBAND	NID OF WIFE	DOCUMENTS
This should list all marriages awaiting approval.		



Dashboard

AH

alvee hawak

Profile Requests Admin

Logout

Find Marriage

NID of Husband

NID of Wife

Find

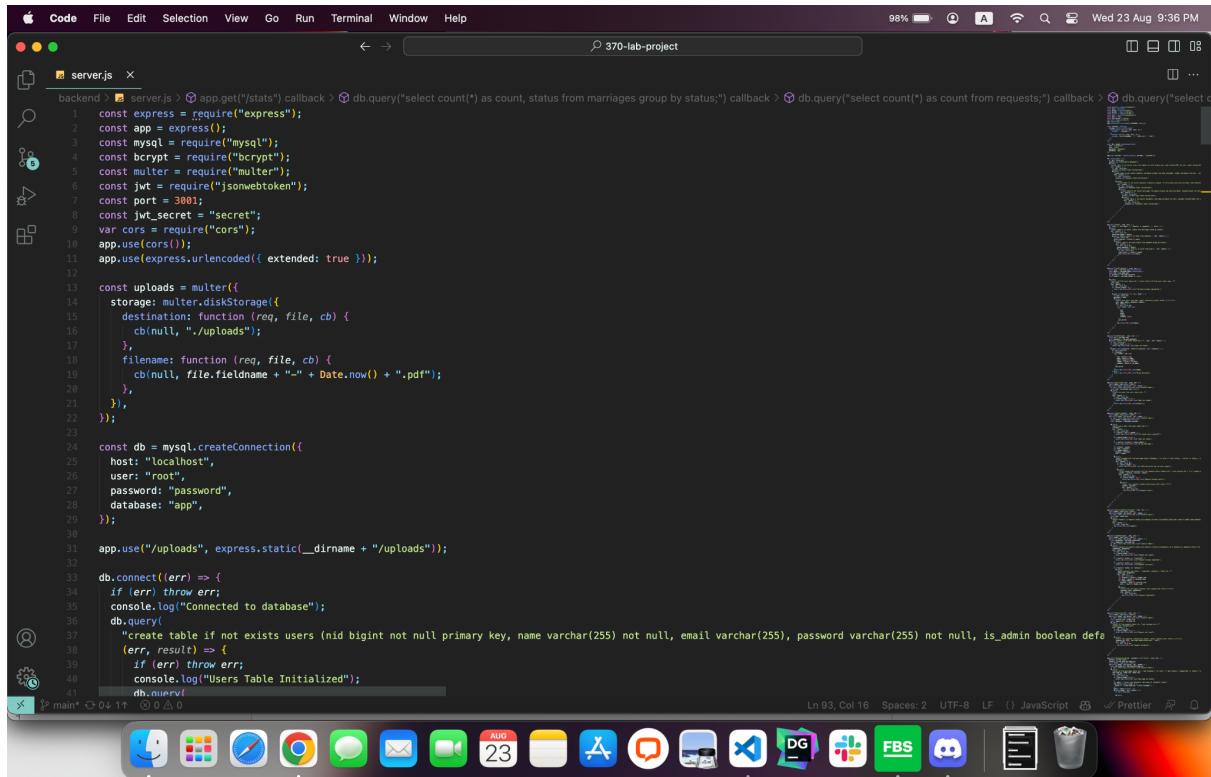


Backend Development

Backend has been coded with expressjs, mysql and other nodejs libraries.

Tables are created if they do not exist the moment the backend startup. Other than basic processing, no extra libraries are used for mysql. Every set of data are queried with raw mysql which can be seen in the source code of the backend. The whole backend is written in a single file. Tables are created with proper foreign keys and datatypes.

Here are some screenshots of the backend:



The screenshot shows a Mac OS X desktop environment. In the foreground, a Code editor window is open, displaying the contents of a file named 'server.js'. The code is written in JavaScript and includes imports for express, mysql, bcrypt, multer, and jwt, along with configuration for port, CORS, and middleware. It also contains logic for connecting to a MySQL database and creating a 'Users' table if it doesn't already exist. The editor interface includes a sidebar with file navigation and a status bar at the bottom indicating the line and column numbers (Ln 93, Col 16), spaces used (Spaces: 2), and the file type (JavaScript). In the background, a web browser window is visible, showing a simple landing page with the text '370-lab-project' and some placeholder content. The desktop dock at the bottom contains icons for various applications like Finder, Mail, and Safari. The system tray shows the date (AUG 23) and battery level (98%).

A screenshot of a macOS desktop environment. At the top is a dark-themed menu bar with the Apple logo and various system and application menus. The main window is a code editor showing a file named 'server.js'. The code is a Node.js script for a database migration or setup. It uses the 'db' object to run several SQL-like queries to create tables for 'users', 'requests', 'payments', 'marriages', and 'documents'. The code includes error handling and logging. The bottom of the screen shows a dock with various application icons, including Finder, Mail, Safari, and others. A status bar at the bottom right displays battery level (98%), signal strength, and the date and time (Wed 23 Aug 9:37 PM).

```
backend > server.js > app.get("/stats") callback > db.query("select count(*) as count, status from marriages group by status;") callback > db.query("select count(*) as count from requests;") callback > db.query("select count(*) as count from users;") callback > db.connect((err) => {
  if (err) throw err;
  console.log("Connected to database");
  db.query(
    "create table if not exists users (nid bigint not null primary key, name varchar(255) not null, email varchar(255), password varchar(255) not null, is_admin boolean default false);
  (err, result) => {
    if (err) throw err;
    console.log("Users Table Initialized");
  }
  db.query(
    "create table if not exists requests (id bigint primary key auto_increment, sender_nid bigint not null , reciever_nid bigint not null , date datetime default now()),
  (err, result) => {
    if (err) throw err;
    console.log("Requests Table Initialized");
  }

  db.query(
    "create table if not exists payments (request_id bigint, id int primary key auto_increment, date datetime default now(), status enum('pending','completed','cancel') default 'pending');
  (err, result) => {
    if (err) throw err;
    console.log("Payments Table Initialized");
  }
  db.query(
    "create table if not exists marriages (id bigint primary key auto_increment, husband bigint not null, wife bigint not null , date datetime DEFAULT now(), status enum('active','inactive') default 'active');
  (err, result) => {
    if (err) throw err;
    console.log("Marriages Table Initialized");
  }
  db.query(
    "create table if not exists documents (marriage_id bigint not null, document varchar(1024) not null, foreign key (marriage_id) references marriages(id));
  (err, result) => {
    if (err) throw err;
    console.log("Documents Table Initialized");
  }
);
```

```
 101 app.post("/auth/register", (req, res) => {
  102   const name = req.body.name.toLowerCase();
  103   const email = req.body.email.toLowerCase();
  104   const nid = req.body.nid;
  105   var password = req.body.password;
  106   const isMale = req.body.isMale == "male";
  107
  108   db.query(
  109     "select nid from users where nid = ? union select nid from users where name = ?",
  110     [nid, name]
  111   ).then(result) => {
  112     if (err) throw err;
  113     if (result.length > 0) {
  114       return res.status(400).send("You have already registered.");
  115     }
  116
  117     bcrypt.hash(password, 10, (err, hash) => {
  118       if (err) throw err;
  119       password = hash;
  120       db.query(
  121         "insert into users (nid,name, email, password, is_male) values (?, ?, ?, ?, ?)",
  122         [nid, name, email, password, isMale]
  123       ).then(result) => {
  124         if (err) throw err;
  125         const token = jwt.sign(
  126           {
  127             nid,
  128             name,
  129             email,
  130             isMale,
  131             isAdmin: false,
  132           },
  133           jwt_secret
  134         );
  135         res.status(200).send(token);
  136       });
  137     });
  138   });
  139 });
  140
  141 });
  142
  143 app.post("/auth/login", (req, res) => {
  144   const nid = req.body.nid;
  145   const password = req.body.password;
  146   db.query("select * from users where nid = ?", [nid], (err, result) => {
  147     if (err) throw err;
  148     if (result.length === 0) {
  149       return res.status(400).send("User not found");
  150     }
  151     bcrypt.compare(password, result[0].password, (err, response) => {
  152       if (err) throw err;
  153       if (response) {
  154         const token = jwt.sign(
  155           {
  156             nid: result[0].nid,
  157             name: result[0].name,
  158             email: result[0].email,
  159             isMale: result[0].is_male,
  160             isAdmin: result[0].is_admin,
  161           },
  162           jwt_secret
  163         );
  164         return res.status(200).send(token);
  165       } else {
  166         return res.status(400).send("Wrong Password");
  167       }
  168     });
  169   });
  170 });
  171
  172 app.get("/marry/usernid", (req, res) => {
  173   const token = req.header("token");
  174   jwt.verify(token, jwt_secret, (err, token) => {
  175     if (err) return res.status(403).send("Invalid Token");
  176     const nid = req.params.nid.slice(1);
  177     db.query(
  178       "select nid, name from users where nid = ?",
  179       ...
  180     );
  181   });
  182 });
  183
  184 app.get("/marry/nid", (req, res) => {
  185   const token = req.header("token");
  186   jwt.verify(token, jwt_secret, (err, token) => {
  187     if (err) return res.status(403).send("Invalid Token");
  188     const nid = req.params.nid;
  189     db.query(
  190       "select name from users where nid = ?",
  191       ...
  192     );
  193   });
  194 });
  195
  196 app.get("/marry/names", (req, res) => {
  197   const token = req.header("token");
  198   jwt.verify(token, jwt_secret, (err, token) => {
  199     if (err) return res.status(403).send("Invalid Token");
  200     const names = [
  201       "John Doe",
  202       "Jane Smith",
  203       "Mike Johnson",
  204       "Sarah Williams",
  205       "David Lee",
  206       "Emily Davis",
  207       "Robert Wilson",
  208       "Sarah Thompson",
  209       "Michael Brown",
  210       "Laura Green",
  211       "Christopher White",
  212       "Sarah Lee",
  213       "David Thompson",
  214       "Sarah Wilson",
  215       "Michael Green",
  216       "Laura White",
  217       "Christopher Lee",
  218       "Sarah Brown",
  219       "David Green",
  220       "Sarah Wilson",
  221       "Michael Green",
  222       "Laura White",
  223       "Christopher Lee",
  224       "Sarah Brown",
  225       "David Green",
  226       "Sarah Wilson",
  227       "Michael Green",
  228       "Laura White",
  229       "Christopher Lee",
  230       "Sarah Brown",
  231       "David Green",
  232       "Sarah Wilson",
  233       "Michael Green",
  234       "Laura White",
  235       "Christopher Lee",
  236       "Sarah Brown",
  237       "David Green",
  238       "Sarah Wilson",
  239       "Michael Green",
  240       "Laura White",
  241       "Christopher Lee",
  242       "Sarah Brown",
  243       "David Green",
  244       "Sarah Wilson",
  245       "Michael Green",
  246       "Laura White",
  247       "Christopher Lee",
  248       "Sarah Brown",
  249       "David Green",
  250       "Sarah Wilson",
  251       "Michael Green",
  252       "Laura White",
  253       "Christopher Lee",
  254       "Sarah Brown",
  255       "David Green",
  256       "Sarah Wilson",
  257       "Michael Green",
  258       "Laura White",
  259       "Christopher Lee",
  260       "Sarah Brown",
  261       "David Green",
  262       "Sarah Wilson",
  263       "Michael Green",
  264       "Laura White",
  265       "Christopher Lee",
  266       "Sarah Brown",
  267       "David Green",
  268       "Sarah Wilson",
  269       "Michael Green",
  270       "Laura White",
  271       "Christopher Lee",
  272       "Sarah Brown",
  273       "David Green",
  274       "Sarah Wilson",
  275       "Michael Green",
  276       "Laura White",
  277       "Christopher Lee",
  278       "Sarah Brown",
  279       "David Green",
  280       "Sarah Wilson",
  281       "Michael Green",
  282       "Laura White",
  283       "Christopher Lee",
  284       "Sarah Brown",
  285       "David Green",
  286       "Sarah Wilson",
  287       "Michael Green",
  288       "Laura White",
  289       "Christopher Lee",
  290       "Sarah Brown",
  291       "David Green",
  292       "Sarah Wilson",
  293       "Michael Green",
  294       "Laura White",
  295       "Christopher Lee",
  296       "Sarah Brown",
  297       "David Green",
  298       "Sarah Wilson",
  299       "Michael Green",
  300       "Laura White",
  301       "Christopher Lee",
  302       "Sarah Brown",
  303       "David Green",
  304       "Sarah Wilson",
  305       "Michael Green",
  306       "Laura White",
  307       "Christopher Lee",
  308       "Sarah Brown",
  309       "David Green",
  310       "Sarah Wilson",
  311       "Michael Green",
  312       "Laura White",
  313       "Christopher Lee",
  314       "Sarah Brown",
  315       "David Green",
  316       "Sarah Wilson",
  317       "Michael Green",
  318       "Laura White",
  319       "Christopher Lee",
  320       "Sarah Brown",
  321       "David Green",
  322       "Sarah Wilson",
  323       "Michael Green",
  324       "Laura White",
  325       "Christopher Lee",
  326       "Sarah Brown",
  327       "David Green",
  328       "Sarah Wilson",
  329       "Michael Green",
  330       "Laura White",
  331       "Christopher Lee",
  332       "Sarah Brown",
  333       "David Green",
  334       "Sarah Wilson",
  335       "Michael Green",
  336       "Laura White",
  337       "Christopher Lee",
  338       "Sarah Brown",
  339       "David Green",
  340       "Sarah Wilson",
  341       "Michael Green",
  342       "Laura White",
  343       "Christopher Lee",
  344       "Sarah Brown",
  345       "David Green",
  346       "Sarah Wilson",
  347       "Michael Green",
  348       "Laura White",
  349       "Christopher Lee",
  350       "Sarah Brown",
  351       "David Green",
  352       "Sarah Wilson",
  353       "Michael Green",
  354       "Laura White",
  355       "Christopher Lee",
  356       "Sarah Brown",
  357       "David Green",
  358       "Sarah Wilson",
  359       "Michael Green",
  360       "Laura White",
  361       "Christopher Lee",
  362       "Sarah Brown",
  363       "David Green",
  364       "Sarah Wilson",
  365       "Michael Green",
  366       "Laura White",
  367       "Christopher Lee",
  368       "Sarah Brown",
  369       "David Green",
  370       "Sarah Wilson",
  371       "Michael Green",
  372       "Laura White",
  373       "Christopher Lee",
  374       "Sarah Brown",
  375       "David Green",
  376       "Sarah Wilson",
  377       "Michael Green",
  378       "Laura White",
  379       "Christopher Lee",
  380       "Sarah Brown",
  381       "David Green",
  382       "Sarah Wilson",
  383       "Michael Green",
  384       "Laura White",
  385       "Christopher Lee",
  386       "Sarah Brown",
  387       "David Green",
  388       "Sarah Wilson",
  389       "Michael Green",
  390       "Laura White",
  391       "Christopher Lee",
  392       "Sarah Brown",
  393       "David Green",
  394       "Sarah Wilson",
  395       "Michael Green",
  396       "Laura White",
  397       "Christopher Lee",
  398       "Sarah Brown",
  399       "David Green",
  400       "Sarah Wilson",
  401       "Michael Green",
  402       "Laura White",
  403       "Christopher Lee",
  404       "Sarah Brown",
  405       "David Green",
  406       "Sarah Wilson",
  407       "Michael Green",
  408       "Laura White",
  409       "Christopher Lee",
  410       "Sarah Brown",
  411       "David Green",
  412       "Sarah Wilson",
  413       "Michael Green",
  414       "Laura White",
  415       "Christopher Lee",
  416       "Sarah Brown",
  417       "David Green",
  418       "Sarah Wilson",
  419       "Michael Green",
  420       "Laura White",
  421       "Christopher Lee",
  422       "Sarah Brown",
  423       "David Green",
  424       "Sarah Wilson",
  425       "Michael Green",
  426       "Laura White",
  427       "Christopher Lee",
  428       "Sarah Brown",
  429       "David Green",
  430       "Sarah Wilson",
  431       "Michael Green",
  432       "Laura White",
  433       "Christopher Lee",
  434       "Sarah Brown",
  435       "David Green",
  436       "Sarah Wilson",
  437       "Michael Green",
  438       "Laura White",
  439       "Christopher Lee",
  440       "Sarah Brown",
  441       "David Green",
  442       "Sarah Wilson",
  443       "Michael Green",
  444       "Laura White",
  445       "Christopher Lee",
  446       "Sarah Brown",
  447       "David Green",
  448       "Sarah Wilson",
  449       "Michael Green",
  450       "Laura White",
  451       "Christopher Lee",
  452       "Sarah Brown",
  453       "David Green",
  454       "Sarah Wilson",
  455       "Michael Green",
  456       "Laura White",
  457       "Christopher Lee",
  458       "Sarah Brown",
  459       "David Green",
  460       "Sarah Wilson",
  461       "Michael Green",
  462       "Laura White",
  463       "Christopher Lee",
  464       "Sarah Brown",
  465       "David Green",
  466       "Sarah Wilson",
  467       "Michael Green",
  468       "Laura White",
  469       "Christopher Lee",
  470       "Sarah Brown",
  471       "David Green",
  472       "Sarah Wilson",
  473       "Michael Green",
  474       "Laura White",
  475       "Christopher Lee",
  476       "Sarah Brown",
  477       "David Green",
  478       "Sarah Wilson",
  479       "Michael Green",
  480       "Laura White",
  481       "Christopher Lee",
  482       "Sarah Brown",
  483       "David Green",
  484       "Sarah Wilson",
  485       "Michael Green",
  486       "Laura White",
  487       "Christopher Lee",
  488       "Sarah Brown",
  489       "David Green",
  490       "Sarah Wilson",
  491       "Michael Green",
  492       "Laura White",
  493       "Christopher Lee",
  494       "Sarah Brown",
  495       "David Green",
  496       "Sarah Wilson",
  497       "Michael Green",
  498       "Laura White",
  499       "Christopher Lee",
  500       "Sarah Brown",
  501       "David Green",
  502       "Sarah Wilson",
  503       "Michael Green",
  504       "Laura White",
  505       "Christopher Lee",
  506       "Sarah Brown",
  507       "David Green",
  508       "Sarah Wilson",
  509       "Michael Green",
  510       "Laura White",
  511       "Christopher Lee",
  512       "Sarah Brown",
  513       "David Green",
  514       "Sarah Wilson",
  515       "Michael Green",
  516       "Laura White",
  517       "Christopher Lee",
  518       "Sarah Brown",
  519       "David Green",
  520       "Sarah Wilson",
  521       "Michael Green",
  522       "Laura White",
  523       "Christopher Lee",
  524       "Sarah Brown",
  525       "David Green",
  526       "Sarah Wilson",
  527       "Michael Green",
  528       "Laura White",
  529       "Christopher Lee",
  530       "Sarah Brown",
  531       "David Green",
  532       "Sarah Wilson",
  533       "Michael Green",
  534       "Laura White",
  535       "Christopher Lee",
  536       "Sarah Brown",
  537       "David Green",
  538       "Sarah Wilson",
  539       "Michael Green",
  540       "Laura White",
  541       "Christopher Lee",
  542       "Sarah Brown",
  543       "David Green",
  544       "Sarah Wilson",
  545       "Michael Green",
  546       "Laura White",
  547       "Christopher Lee",
  548       "Sarah Brown",
  549       "David Green",
  550       "Sarah Wilson",
  551       "Michael Green",
  552       "Laura White",
  553       "Christopher Lee",
  554       "Sarah Brown",
  555       "David Green",
  556       "Sarah Wilson",
  557       "Michael Green",
  558       "Laura White",
  559       "Christopher Lee",
  560       "Sarah Brown",
  561       "David Green",
  562       "Sarah Wilson",
  563       "Michael Green",
  564       "Laura White",
  565       "Christopher Lee",
  566       "Sarah Brown",
  567       "David Green",
  568       "Sarah Wilson",
  569       "Michael Green",
  570       "Laura White",
  571       "Christopher Lee",
  572       "Sarah Brown",
  573       "David Green",
  574       "Sarah Wilson",
  575       "Michael Green",
  576       "Laura White",
  577       "Christopher Lee",
  578       "Sarah Brown",
  579       "David Green",
  580       "Sarah Wilson",
  581       "Michael Green",
  582       "Laura White",
  583       "Christopher Lee",
  584       "Sarah Brown",
  585       "David Green",
  586       "Sarah Wilson",
  587       "Michael Green",
  588       "Laura White",
  589       "Christopher Lee",
  590       "Sarah Brown",
  591       "David Green",
  592       "Sarah Wilson",
  593       "Michael Green",
  594       "Laura White",
  595       "Christopher Lee",
  596       "Sarah Brown",
  597       "David Green",
  598       "Sarah Wilson",
  599       "Michael Green",
  600       "Laura White",
  601       "Christopher Lee",
  602       "Sarah Brown",
  603       "David Green",
  604       "Sarah Wilson",
  605       "Michael Green",
  606       "Laura White",
  607       "Christopher Lee",
  608       "Sarah Brown",
  609       "David Green",
  610       "Sarah Wilson",
  611       "Michael Green",
  612       "Laura White",
  613       "Christopher Lee",
  614       "Sarah Brown",
  615       "David Green",
  616       "Sarah Wilson",
  617       "Michael Green",
  618       "Laura White",
  619       "Christopher Lee",
  620       "Sarah Brown",
  621       "David Green",
  622       "Sarah Wilson",
  623       "Michael Green",
  624       "Laura White",
  625       "Christopher Lee",
  626       "Sarah Brown",
  627       "David Green",
  628       "Sarah Wilson",
  629       "Michael Green",
  630       "Laura White",
  631       "Christopher Lee",
  632       "Sarah Brown",
  633       "David Green",
  634       "Sarah Wilson",
  635       "Michael Green",
  636       "Laura White",
  637       "Christopher Lee",
  638       "Sarah Brown",
  639       "David Green",
  640       "Sarah Wilson",
  641       "Michael Green",
  642       "Laura White",
  643       "Christopher Lee",
  644       "Sarah Brown",
  645       "David Green",
  646       "Sarah Wilson",
  647       "Michael Green",
  648       "Laura White",
  649       "Christopher Lee",
  650       "Sarah Brown",
  651       "David Green",
  652       "Sarah Wilson",
  653       "Michael Green",
  654       "Laura White",
  655       "Christopher Lee",
  656       "Sarah Brown",
  657       "David Green",
  658       "Sarah Wilson",
  659       "Michael Green",
  660       "Laura White",
  661       "Christopher Lee",
  662       "Sarah Brown",
  663       "David Green",
  664       "Sarah Wilson",
  665       "Michael Green",
  666       "Laura White",
  667       "Christopher Lee",
  668       "Sarah Brown",
  669       "David Green",
  670       "Sarah Wilson",
  671       "Michael Green",
  672       "Laura White",
  673       "Christopher Lee",
  674       "Sarah Brown",
  675       "David Green",
  676       "Sarah Wilson",
  677       "Michael Green",
  678       "Laura White",
  679       "Christopher Lee",
  680       "Sarah Brown",
  681       "David Green",
  682       "Sarah Wilson",
  683       "Michael Green",
  684       "Laura White",
  685       "Christopher Lee",
  686       "Sarah Brown",
  687       "David Green",
  688       "Sarah Wilson",
  689       "Michael Green",
  690       "Laura White",
  691       "Christopher Lee",
  692       "Sarah Brown",
  693       "David Green",
  694       "Sarah Wilson",
  695       "Michael Green",
  696       "Laura White",
  697       "Christopher Lee",
  698       "Sarah Brown",
  699       "David Green",
  700       "Sarah Wilson",
  701       "Michael Green",
  702       "Laura White",
  703       "Christopher Lee",
  704       "Sarah Brown",
  705       "David Green",
  706       "Sarah Wilson",
  707       "Michael Green",
  708       "Laura White",
  709       "Christopher Lee",
  710       "Sarah Brown",
  711       "David Green",
  712       "Sarah Wilson",
  713       "Michael Green",
  714       "Laura White",
  715       "Christopher Lee",
  716       "Sarah Brown",
  717       "David Green",
  718       "Sarah Wilson",
  719       "Michael Green",
  720       "Laura White",
  721       "Christopher Lee",
  722       "Sarah Brown",
  723       "David Green",
  724       "Sarah Wilson",
  725       "Michael Green",
  726       "Laura White",
  727       "Christopher Lee",
  728       "Sarah Brown",
  729       "David Green",
  730       "Sarah Wilson",
  731       "Michael Green",
  732       "Laura White",
  733       "Christopher Lee",
  734       "Sarah Brown",
  735       "David Green",
  736       "Sarah Wilson",
  737       "Michael Green",
  738       "Laura White",
  739       "Christopher Lee",
  740       "Sarah Brown",
  741       "David Green",
  742       "Sarah Wilson",
  743       "Michael Green",
  744       "Laura White",
  745       "Christopher Lee",
  746       "Sarah Brown",
  747       "David Green",
  748       "Sarah Wilson",
  749       "Michael Green",
  750       "Laura White",
  751       "Christopher Lee",
  752       "Sarah Brown",
  753       "David Green",
  754       "Sarah Wilson",
  755       "Michael Green",
  756       "Laura White",
  757       "Christopher Lee",
  758       "Sarah Brown",
  759       "David Green",
  760       "Sarah Wilson",
  761       "Michael Green",
  762       "Laura White",
  763       "Christopher Lee",
  764       "Sarah Brown",
  765       "David Green",
  766       "Sarah Wilson",
  767       "Michael Green",
  768       "Laura White",
  769       "Christopher Lee",
  770       "Sarah Brown",
  771       "David Green",
  772       "Sarah Wilson",
  773       "Michael Green",
  774       "Laura White",
  775       "Christopher Lee",
  776       "Sarah Brown",
  777       "David Green",
  778       "Sarah Wilson",
  779       "Michael Green",
  780       "Laura White",
  781       "Christopher Lee",
  782       "Sarah Brown",
  783       "David Green",
  784       "Sarah Wilson",
  785       "Michael Green",
  786       "Laura White",
  787       "Christopher Lee",
  788       "Sarah Brown",
  789       "David Green",
  790       "Sarah Wilson",
  791       "Michael Green",
  792       "Laura White",
  793       "Christopher Lee",
  794       "Sarah Brown",
  795       "David Green",
  796       "Sarah Wilson",
  797       "Michael Green",
  798       "Laura White",
  799       "Christopher Lee",
  800       "Sarah Brown",
  801       "David Green",
  802       "Sarah Wilson",
  803       "Michael Green",
  804       "Laura White",
  805       "Christopher Lee",
  806       "Sarah Brown",
  807       "David Green",
  808       "Sarah Wilson",
  809       "Michael Green",
  810       "Laura White",
  811       "Christopher Lee",
  812       "Sarah Brown",
  813       "David Green",
  814       "Sarah Wilson",
  815       "Michael Green",
  816       "Laura White",
  817       "Christopher Lee",
  818       "Sarah Brown",
  819       "David Green",
  820       "Sarah Wilson",
  821       "Michael Green",
  822       "Laura White",
  823       "Christopher Lee",
  824       "Sarah Brown",
  825       "David Green",
  826       "Sarah Wilson",
  827       "Michael Green",
  828       "Laura White",
  829       "Christopher Lee",
  830       "Sarah Brown",
  831       "David Green",
  832       "Sarah Wilson",
  833       "Michael Green",
  834       "Laura White",
  835       "Christopher Lee",
  836       "Sarah Brown",
  837       "David Green",
  838       "Sarah Wilson",
  839       "Michael Green",
  840       "Laura White",
  841       "Christopher Lee",
  842       "Sarah Brown",
  843       "David Green",
  844       "Sarah Wilson",
  845       "Michael Green",
  846       "Laura White",
  847       "Christopher Lee",
  848       "Sarah Brown",
  849       "David Green",
  850       "Sarah Wilson",
  851       "Michael Green",
  852       "Laura White",
  853       "Christopher Lee",
  854       "Sarah Brown",
  855       "David Green",
  856       "Sarah Wilson",
  857       "Michael Green",
  858       "Laura White",
  859       "Christopher Lee",
  860       "Sarah Brown",
  861       "David Green",
  862       "Sarah Wilson",
  863       "Michael Green",
  864       "Laura White",
  865       "Christopher Lee",
  866       "Sarah Brown",
  867       "David Green",
  868       "Sarah Wilson",
  869       "Michael Green",
  870       "Laura White",
  871       "Christopher Lee",
  872       "Sarah Brown",
  873       "David Green",
  874       "Sarah Wilson",
  875       "Michael Green",
  876       "Laura White",
  877       "Christopher Lee",
  878       "Sarah Brown",
  879       "David Green",
  880       "Sarah Wilson",
  881       "Michael Green",
  882       "Laura White",
  883       "Christopher Lee",
  884       "Sarah Brown",
  885       "David Green",
  886       "Sarah Wilson",
  887       "Michael Green",
  888       "Laura White",
  889       "Christopher Lee",
  890       "Sarah Brown",
  891       "David Green",
  892       "Sarah Wilson",
  893       "Michael Green",
  894       "Laura White",
  895       "Christopher Lee",
  896       "Sarah Brown",
  897       "David Green",
  898       "Sarah Wilson",
  899       "Michael Green",
  900       "Laura White",
  901       "Christopher Lee",
  902       "Sarah Brown",
  903       "David Green",
  904       "Sarah Wilson",
  905       "Michael Green",
  906       "Laura White",
  907       "Christopher Lee",
  908       "Sarah Brown",
  909       "David Green",
  910       "Sarah Wilson",
  911       "Michael Green",
  912       "Laura White",
  913       "Christopher Lee",
  914       "Sarah Brown",
  915       "David Green",
  916       "Sarah Wilson",
  917       "Michael Green",
  918       "Laura White",
  919       "Christopher Lee",
  920       "Sarah Brown",
  921       "David Green",
  922       "Sarah Wilson",
  923       "Michael Green",
  924       "Laura White",
  925       "Christopher Lee",
  926       "Sarah Brown",
  927       "David Green",
  928       "Sarah Wilson",
  929       "Michael Green",
  930       "Laura White",
  931       "Christopher Lee",
  932       "Sarah Brown",
  933       "David Green",
  934       "Sarah Wilson",
  935       "Michael Green",
  936       "Laura White",
  937       "Christopher Lee",
  938       "Sarah Brown",
  939       "David Green",
  940       "Sarah Wilson",
  941       "Michael Green",
  942       "Laura White",
  943       "Christopher Lee",
  944       "Sarah Brown",
  945       "David Green",
  946       "Sarah Wilson",
  947       "Michael Green",
  948       "Laura White",
  949       "Christopher Lee",
  950       "Sarah Brown",
  951       "David Green",
  952       "Sarah Wilson",
  953       "Michael Green",
  954       "Laura White",
  955       "Christopher Lee",
  956       "Sarah Brown",
  957       "David Green",
  958       "Sarah Wilson",
  959       "Michael Green",
  960       "Laura White",
  961       "Christopher Lee",
  962       "Sarah Brown",
  963       "David Green",
  964       "Sarah Wilson",
  965       "Michael Green",
  966       "Laura White",
  967       "Christopher Lee",
  968       "Sarah Brown",
  969       "David Green",
  970       "Sarah Wilson",
  971       "Michael Green",
  972       "Laura White",
  973       "Christopher Lee",
  974       "Sarah Brown",
  975       "David Green",
  976       "Sarah Wilson",
  977       "Michael Green",
  978       "Laura White",
  979       "Christopher Lee",
  980       "Sarah Brown",
  981       "David Green",
  982       "Sarah Wilson",
  983       "Michael Green",
  984       "Laura White",
  985       "Christopher Lee",
  986       "Sarah Brown",
  987       "David Green",
  988       "Sarah Wilson",
  989       "Michael Green",
  990       "Laura White",
  991       "Christopher Lee",
  992       "Sarah Brown",
  993       "David Green",
  994       "Sarah Wilson",
  995       "Michael Green",
  996       "Laura White",
  997       "Christopher Lee",
  998       "Sarah Brown",
  999       "David Green",
  1000      "Sarah Wilson",
  1001      "Michael Green",
  1002      "Laura White",
  1003      "Christopher Lee",
  1004      "Sarah Brown",
  1005      "David Green",
  1006      "Sarah Wilson",
  1007      "Michael Green",
  1008      "Laura White",
  1009      "Christopher Lee",
  1010      "Sarah Brown",
  1011      "David Green",
  1012      "Sarah Wilson",
  1013      "Michael Green",
  1014      "Laura White",
  1015      "Christopher Lee",
  1016      "Sarah Brown",
  1017      "David Green",
  1018      "Sarah Wilson",
  1019      "Michael Green",
  1020      "Laura White",
  1021      "Christopher Lee",
  1022      "Sarah Brown",
  1023      "David Green",
  1024      "Sarah Wilson",
  1025      "Michael Green",
  1026      "Laura White",
  1027      "Christopher Lee",
  1028      "Sarah Brown",
  1029      "David Green",
  1030      "Sarah Wilson",
  1031      "Michael Green",
  1032      "Laura White",
  1033      "Christopher Lee",
  1034      "Sarah Brown",
  1035      "David Green",
  1036      "Sarah Wilson",
  1037      "Michael Green",
  1038      "Laura White",
  1039      "Christopher Lee",
  1040      "Sarah Brown",
  1041      "David Green",
  1042      "Sarah Wilson",
  1043      "Michael Green",
  1044      "Laura White",
  1045      "Christopher Lee",
  1046      "Sarah Brown",
  1047      "David Green",
  1048      "Sarah Wilson",
  1049      "Michael Green",
  1050      "Laura White",
  1051      "Christopher Lee",
  1052      "Sarah Brown",
  1053      "David Green",
  1054      "Sarah Wilson",
  1055      "Michael Green",
  1056      "Laura White",
  1057      "Christopher Lee",
  1058      "Sarah Brown",
  1059      "David Green",
  1060      "Sarah Wilson",
  1061      "Michael Green",
  1062      "Laura White",
  1063      "Christopher Lee",
  1064      "Sarah Brown",
  1065      "David Green",
  1066      "Sarah Wilson",
  1067      "Michael Green",
  1068      "Laura White",
  1069      "Christopher Lee",
  1070      "Sarah Brown",
  1071      "David Green",
  1072      "Sarah Wilson",
  1073      "Michael Green",
  1074      "Laura White",
  1075      "Christopher Lee",
  1076      "Sarah Brown",
  1077      "David Green",
  1078      "Sarah Wilson",
  1079      "Michael Green",
  1080      "Laura White",
  1081      "Christopher Lee",
  1082      "Sarah Brown",
  1083      "David Green",
  1084      "Sarah Wilson",
  1085      "Michael Green",
  1086      "Laura White",
  1087      "Christopher Lee",
  1088      "Sarah Brown",
  1089      "David Green",
  1090      "Sarah Wilson",
  1091      "Michael Green",
  1092      "Laura White",
  1093      "Christopher Lee",
  1094      "Sarah Brown",
  1095      "David Green",
  1096      "Sarah Wilson",
  1097      "Michael Green",
  1098      "Laura White",
  1099      "Christopher Lee",
  1100      "Sarah Brown",
  1101      "David Green",
  1102      "Sarah Wilson",
  1103      "Michael Green",
  1104      "Laura White",
  1105      "Christopher Lee",
  1106      "Sarah Brown",
  1107      "David Green",
  1108      "Sarah Wilson",
  1109      "Michael Green",
  1110      "Laura White",
  1111      "Christopher Lee",
  1112      "Sarah Brown",
  1113      "David Green",
  1114      "Sarah Wilson",
  1115      "Michael Green",
  1116      "Laura White",
  1117      "Christopher Lee",
  1118      "Sarah Brown",
  1119      "David Green",
  1120      "Sarah Wilson",
  1121      "Michael Green",
  1122      "Laura White",
  1123      "Christopher Lee",
  1124      "Sarah Brown",
  1125      "David Green",
  1126      "Sarah Wilson",
  1127      "Michael Green",
  1128      "Laura White",
  1129      "Christopher Lee",
  1130      "Sarah Brown",
  1131      "David Green",
  1132      "Sarah Wilson",
  1133      "Michael Green",
  1134      "Laura White",
  1135      "Christopher Lee",
  1136      "Sarah Brown",
  1137      "David Green",
  1138      "Sarah Wilson",
  1139      "Michael Green",
  1140      "Laura White",
  1141      "Christopher Lee",
  1142      "Sarah Brown",
  1143      "David Green",
  1144      "Sarah Wilson",
  1145      "Michael Green",
  1146      "Laura White",
  1147      "Christopher Lee",
  1148      "Sarah Brown",
  1149      "David Green",
  1150      "Sarah Wilson",
  1151      "Michael Green",
  1152      "Laura White",
  1153      "Christopher Lee",
  1154      "Sarah Brown",
  1155      "David Green",
  1156      "Sarah Wilson",
  1157      "Michael Green",
  1158      "Laura White",
  1159      "Christopher Lee",
  1160      "Sarah Brown",
  1161      "David Green",
  1162      "Sarah Wilson",
  1163      "Michael Green",
  1164      "Laura White",
  1165      "Christopher Lee",
  1166      "Sarah Brown",
  11
```

A screenshot of a Mac desktop showing a Code editor window for a file named 'server.js'. The window title is '370-lab-project'. The code in the editor is a Node.js script for a backend API. It includes routes for '/stats', '/marry/user/:nid', and '/marry/request'. The '/marry/user/:nid' route checks if a user exists by their ID. The '/marry/request' route checks if two users can marry based on their gender. The editor has syntax highlighting and line numbers. The status bar at the bottom shows 'Ln 93, Col 16' and 'Spaces: 2'. The dock at the bottom contains icons for various applications like Finder, Mail, and Safari.

```
backend > server.js > app.get("/stats") callback > db.query("select count(*) as count, status from marriages group by status;") callback > db.query("select count(*) as count from requests;") callback > db.query("select count(*) as count from users where id = ?") callback > res.status(200).send(result[0].count)
```

```
171
172 app.get("/marry/user/:nid", (req, res) => {
173   const token = req.header("token");
174   jwt.verify(token, jwt_secret, (err, token) => {
175     if (err) return res.status(403).send("Invalid Token");
176     const nid = req.params.nid.slice(1);
177     db.query(
178       "select nid,name from users where nid = ?",
179       [nid],
180       (err, result) => {
181         if (err) throw err;
182         if (result.length == 0) {
183           return res.status(400).send("User not found");
184         }
185         return res.status(200).send(result[0]);
186       }
187     );
188   });
189 })
190
191 app.post("/marry/request", (req, res) => {
192   const token = req.header("token");
193   jwt.verify(token, jwt_secret, (err, token) => {
194     if (err) return res.status(403).send("Invalid Token");
195     const sender = token.nid.toString();
196     const receiver = req.body.receiver;
197
198     db.query(
199       "select nid,is_male from users where nid = ?",
200       [receiver],
201       (err, result) => {
202         if (err) throw err;
203         if (result[0].nid == sender) {
204           return res.status(400).send("You cannot marry yourself!");
205         }
206         if (result[0].length == 0) {
207           return res.status(400).send("User not found");
208         }
209         if (result[0].is_male == token.isMale) {
210           return res.status(400).send("No Gay Marriage");
211         }
212       }
213     );
214   });
215 })
```

The screenshot shows a Mac OS X desktop environment. At the top is the Dock with various application icons. Below the Dock is the desktop background featuring a colorful abstract design. In the center, there is a terminal window with a light gray background and black text, displaying a command-line interface. To the right of the terminal is a web browser window showing a login page with fields for 'Email' and 'Password'. Below the browser is a code editor window titled 'server.js' with dark-themed syntax highlighting. The code in the editor is a Node.js script for a REST API, handling routes for '/stats' and '/payments/pay'. The script uses Express.js, body-parser, and MySQL database queries. The bottom of the screen features the macOS menu bar with system status icons like battery level (98%), signal strength, and volume.

```
backend > server.js > app.get("/stats") callback > db.query("select count(*) as count, status from marriages group by status;") callback > db.query("select count(*) as count from requests;") callback : db.query("select count(*) as count from payments where paid_by = ?") callback > res.json({count: result[0].count, status: result[0].status});
```

```
app.get("/marry/requests/received", (req, res) => {
  const token = req.header("token");
  jwt.verify(token, jwt_secret, (err, token) => {
    if (err) return res.status(403).send("Invalid Token");
    const nid = token.nid;
    db.query(
      "select requests.id,requests.sender_nid,requests.receiver_nid,requests.date,users.name as sender_name,payments.status as payment_status,payments.id as payment_id from [nid],
      (err, result) => {
        if (err) throw err;
        res.status(200).send(result);
      }
    );
  });
});

app.post("/payments/pay", (req, res) => {
  const token = req.header("token");
  jwt.verify(token, jwt_secret, (err, token) => {
    const paymentId = req.body.paymentId;
    const requestId = req.body.id;
    if (err) return res.status(403).send("Invalid Token");
    db.query(
      "select requests.id,requests.sender_nid,requests.receiver_nid,payments.id as payment_id, payments.status from payments inner join requests on requests.id = ? where payment_id = ? and requests.id = ?",
      [requestId, paymentId],
      (err, result) => {
        if (err) throw err;
        if (result.length === 0) {
          return res.status(400).send("Payment not found");
        }
        if (result[0].status === "completed") {
          return res.status(400).send("Payment already completed");
        }
        if (result[0].status === "canceled") {
          return res.status(400).send("Payment canceled");
        }
        if (result[0].status === "pending") {
          db.query(
            "update payments set status = 'completed', paid_by = ? where id = ?",
            [token.nid, paymentId],
            (err, result) => {
              if (err) throw err;
              res.json({
                message: "Payment successful"
              });
            }
          );
        }
      }
    );
  });
});
```

Conclusion

In wrapping up this project, I've gained a ton of valuable insights into the world of web development and technology integration. Working on the marriage registration portal using Node.js, React.js, Express.js, and MySQL has been an incredible learning journey for me.

Dealing with data using MySQL was a big learning curve. Designing the database schema, setting up relationships between tables, and writing queries challenged me to think about data organization and retrieval in a more structured way. I've gained hands-on experience in database management and ensuring data integrity, which are skills that I believe will serve me well in future projects.

References

1. <https://chakra-ui.com/getting-started>
2. <https://expressjs.com/en/starter/hello-world.html>
 3. <https://react.dev/>